Artificial Intelligence

Dr. Rajeev Kumar Gupta
Assistant Professor
Pandit Deendayal Energy University
Gandhinagar, Gujarat

Syllabus

Unit-I: INTRODUCTION TO AI AND SEARCHING

- AI Problems, Intelligent Agents, Problem Formulation, Basic Problem Solving Methods. Search strategies, Uniformed Search Strategies, State-Space Search, Bi-Directional Search, BFS, DFS, Heuristic Search Strategies, Local Search Algorithms, Hill Climbing, Greedy Best First Search, A* Search, Simulated Annealing, Measure of performance and analysis of search algorithms.

Unit –II: KNOWLEDGE REPRESENTATION AND INFERENCE

- Game playing, Knowledge representation using-Predicate logic, Introduction to predicate calculus, Resolution, Use of predicate calculus, Knowledge representation using other logic, Structured representation of knowledge, Production based system, Frame based system, First order logic. Inference in first order logic, propositional Vs. first order inference, unification & lifts forward chaining, Backward chaining, Resolution.

Unit – III: NEURAL NETWORKS

- Characteristics of Neural Networks, Historical Development of Neural Networks Principles. Artificial Neural Networks: Terminology, Models of Neuron, Topology, Basic Learning Laws, Pattern Recognition Problem, Basic Functional Units, Pattern Recognition Tasks by the Functional Units.

Unit –IV: Expert System

- Introduction to Expert systems - Architecture of expert systems, Roles of expert systems - Knowledge Acquisition –Meta knowledge, Heuristics. Example of expert systems - MYCIN, DART, XOON, Expert systems shells, Introduction to Planning.

What is Intelligence?

1) Learning

- The simplest is learning by **trial and error.** For example, a simple computer program for solving mate-in-one chess problems might try moves at random until mate is found. The program might then store the solution with the position so that the next time the computer encountered the same position it would recall the solution.

- This simple memorizing of individual items and procedures—known as **rote learning**—is relatively easy to implement on a computer.

- More challenging is the problem of implementing what is called **generalization.** Generalization involves applying past experience to related new situations.

2) Problem solving

- Problem solving, particularly in artificial intelligence, may be characterized as a **systematic search** through a range of possible actions in order to reach some predefined goal or solution.

- Problem-solving methods divide into **special purpose and general purpose.** A special-purpose method is tailor-made for a particular problem and often exploits very specific features of the situation in which the problem is embedded.

- In contrast, a general-purpose method is applicable to a wide variety of problems.

3) Reasoning

- To reason is to **draw inferences** appropriate to the situation. Inferences are classified as either **deductive or inductive.**
- The most significant difference between these forms of reasoning is that in the deductive case the truth of the premises **guarantees the truth of the conclusion,** whereas in the inductive case the truth of the premise lends support to the **conclusion without giving absolute assurance**.
  - In deductive, “Sachin must be in either the museum or the café. He is not in the café; therefore he is in the museum,”
  - In inductive, “Previous accidents of this sort were caused by instrument failure; therefore this accident was caused by instrument failure.”
- Inductive reasoning is common in **science,** where data are collected and tentative models are developed to describe and predict future behaviour—until the appearance of anomalous data forces the model to be revised.
- Deductive reasoning is common in **mathematics and logic**, where elaborate structures of irrefutable theorems are built up from a small set of basic rules.

4) Perception

- In perception the environment is scanned by means of various sensory organs, real or artificial, and the scene is decomposed into separate objects in various spatial relationships.
- Analysis is complicated by the fact that an object may appear different depending on the angle from which it is viewed, the direction and intensity of illumination in the scene, and how much the object contrasts with the surrounding field.

5) Language

- A language is a system of **signs having meaning by convention**. In this sense, language need not be confined to the spoken word.
  - Traffic signs, for example, form a minilanguage, it being a matter of convention that ⚠ means “hazard ahead” in some countries.

Artificial Intelligence

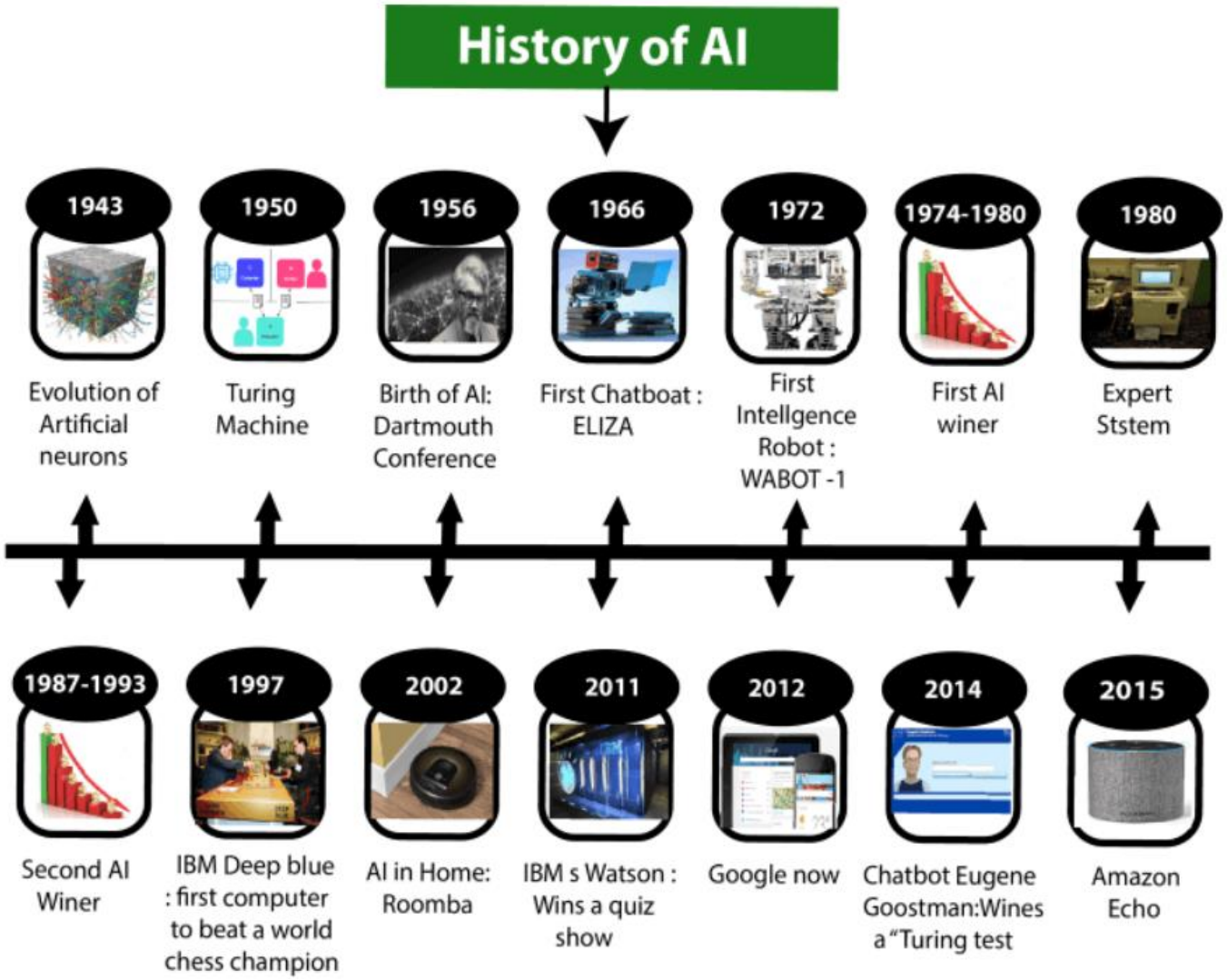
- Artificial intelligence (AI) involves machines that can execute tasks that are similar to that of human intelligence; they were first invented in 1956 by John McCarthy.

- AI, the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

Humanly v/s Rationally

thinking vs. acting		
	Systems that think like humans	Systems that think rationally
	Systems that act like humans	Systems that act rationally Rational Agents

History of Artificial Intelligence



❖Maturation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of **artificial neurons**.

- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called **Hebbian learning**.

- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes **"Computing Machinery and Intelligence"** in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a **Turing test**.

❖The birth of Artificial Intelligence (1952-1956)

- **Year 1955:** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as **"Logic Theorist"**. This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.

- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

❖The golden years-Early enthusiasm (1956-1974)

- **Year 1966:** The researchers emphasized developing algorithms which can solve mathematical problems. W. Joseph created the first chatbot in 1966, which was named as ELIZA.

❖The first AI winter (1974-1980)

- The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.
- During AI winters, an interest of publicity on artificial intelligence was decreased.

❖A boom of AI (1980-1987)

- **Year 1980:** After AI winter duration, AI came back with "**Expert System**". Expert systems were programmed that emulate the decision-making ability of a human expert.
- In the Year 1980, the first national conference of the American Association of Artificial Intelligence **was held at Stanford University**.

❖The second AI winter (1987-1993)

- The duration between the years 1987 to 1993 was the second AI Winter duration.
- Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

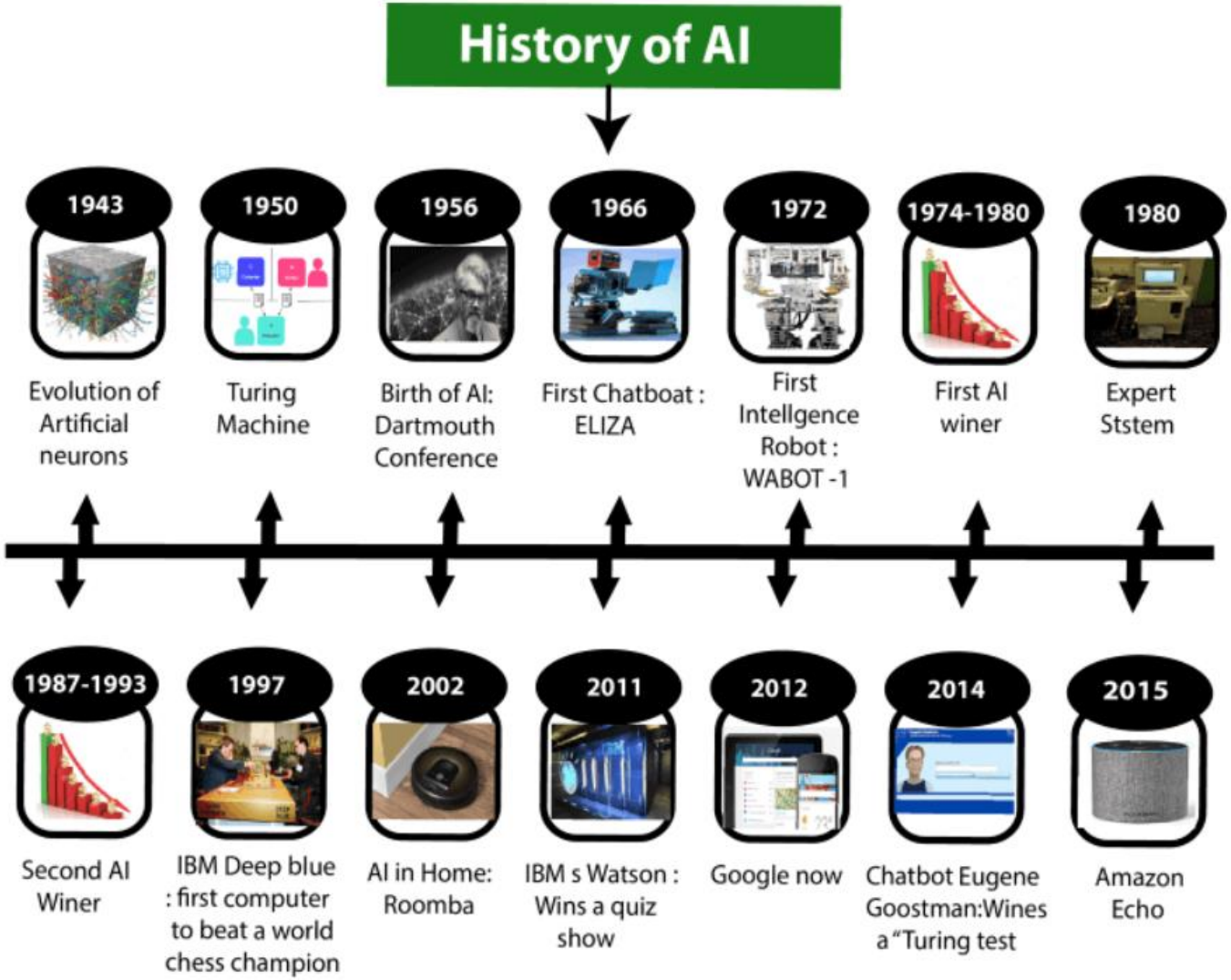
❖The emergence of intelligent agents (1993-2011)

- **Year 1997:** In the year 1997, **IBM Deep Blue** beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
- **Year 2002:** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
- **Year 2006:** AI came in the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.

❖Deep learning, big data and artificial general intelligence (2011-present)

- **Year 2011:** In the year 2011**, IBM's** Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google has launched an **Android app feature "Google now",** which was able to provide information to the user as a prediction.
- **Year 2014:** In the year 2014, **Chatbo**t won a competition in the infamous "Turing test."
- **Year 2018:** The "**Project Debater**" from IBM debated on complex topics with two master debaters and also performed extremely well.
- Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.

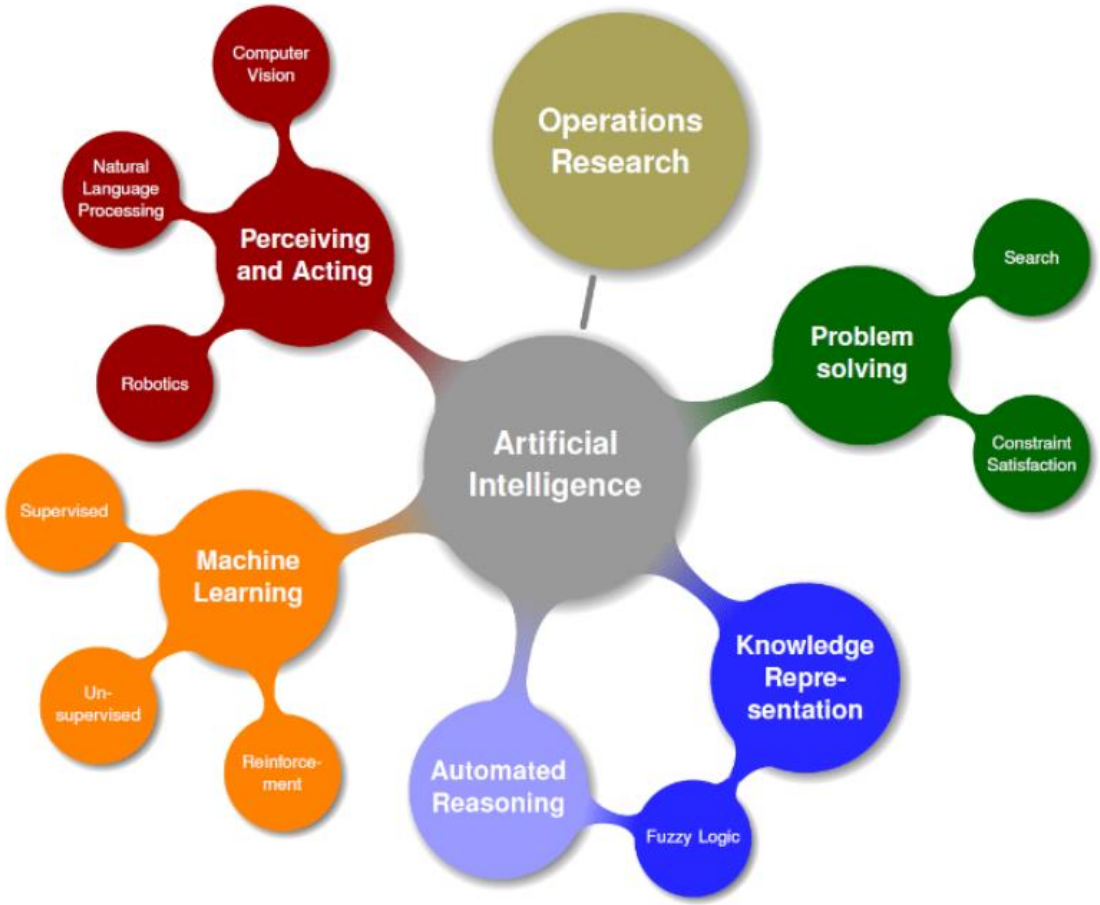
History of Artificial Intelligence



Artificial Intelligence

- AI has three different levels:
  - **Narrow AI**: An artificial intelligence is said to be narrow **when the machine can perform a specific task better than a human**. The current research of AI is here now
  - **General AI**: An artificial intelligence reaches the general state **when it can perform any intellectual task with the same accuracy** level as a human would
  - **Active AI**: An AI is active when it can **beat humans** in many tasks

Different Areas Under Artificial Intelligence



Artificial Intelligence, Machine Learning and Deep Learning

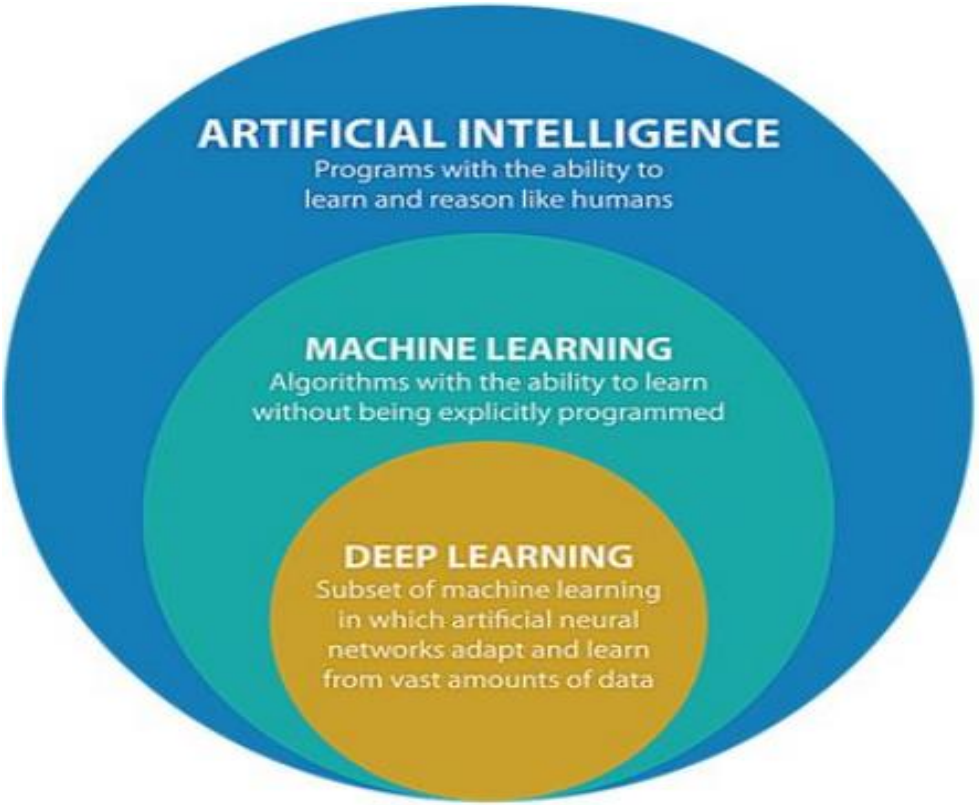


Image Source: https://datacatchup.com/artificial-intelligence-machine-learning-and-deep-learning/

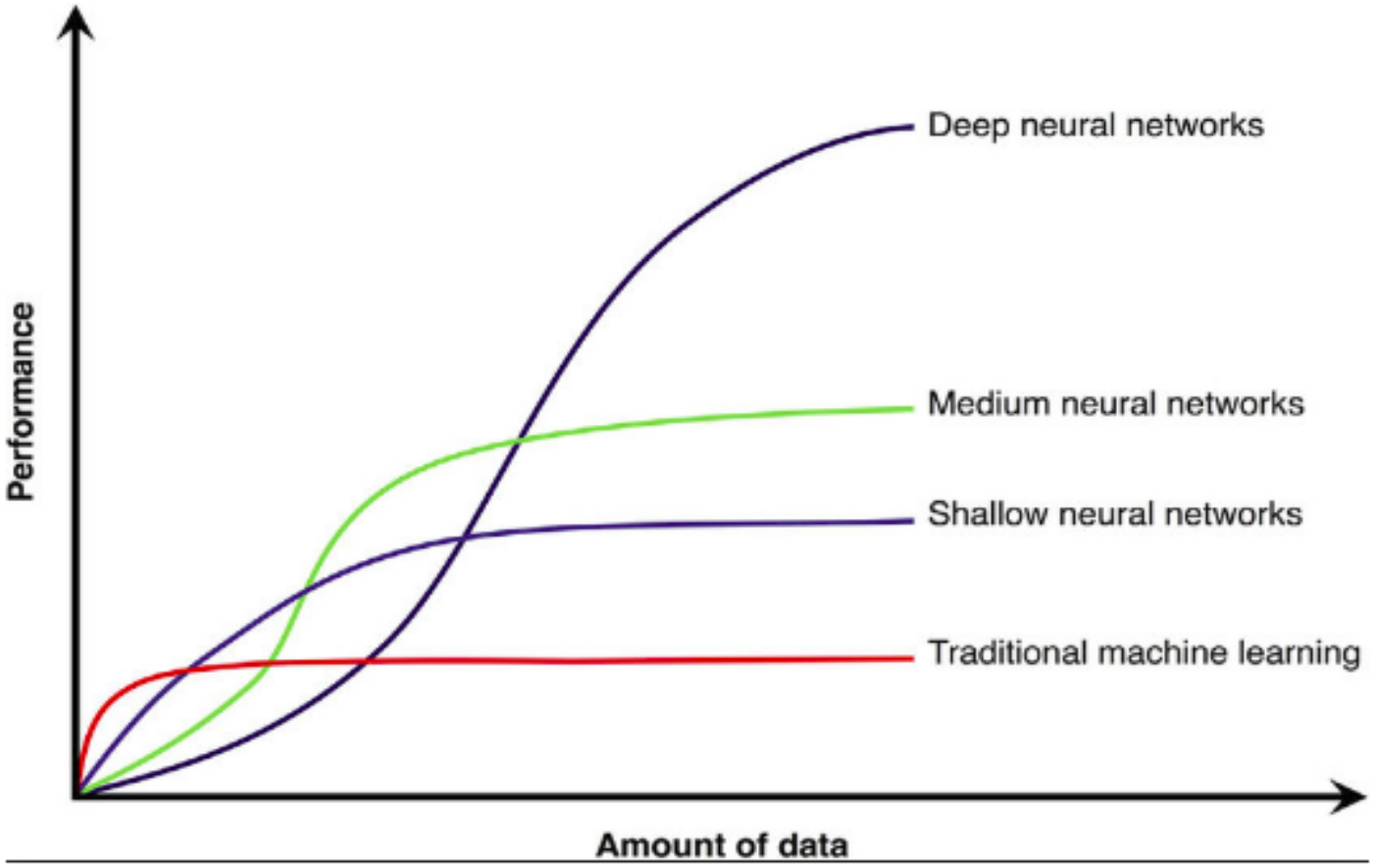
Machine learning

- Machine learning is a simple way of achieving AI. Arthur Samuel summoned the phrase not long after AI, in 1960, defining it as, “the capability to train without being overtly programmed”.

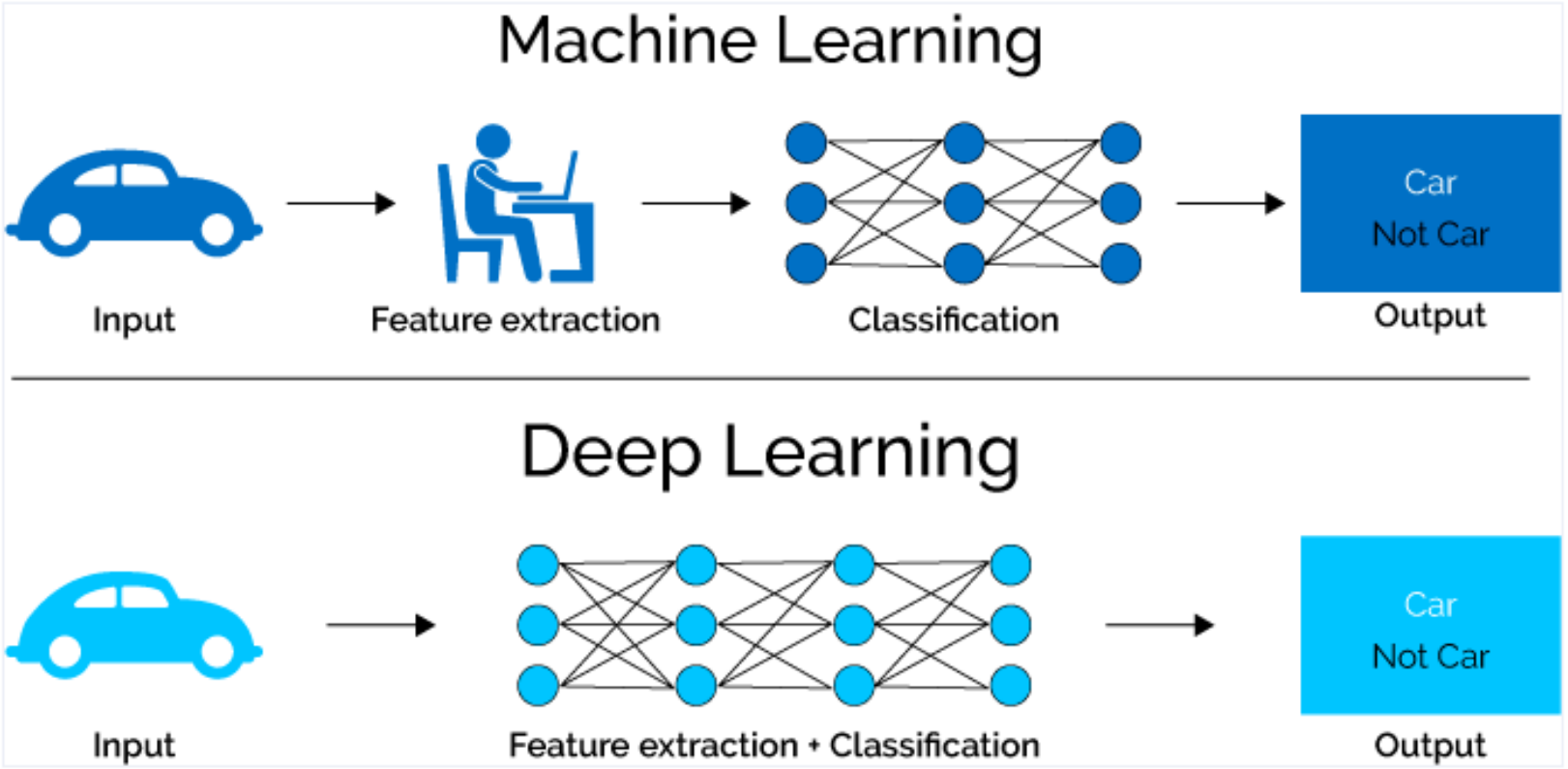
Deep learning

- It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. Deep learning is a computer software that mimics the network of neurons in a brain.
- In deep learning, the learning phase is done through a neural network. A neural network is an architecture where the layers are stacked on top of each other

Machine Learning and Deep Learning Performance w.r.t. Data



Machine Learning v/s Deep Learning



When to use ML or DL?

	Machine learning	Deep learning
Training dataset	Small	Large
Choose features	Yes	No
Number of algorithms	Many	Few
Training time	Short	Long

What are the Most Popular Languages for Machine Learning

- **Python**
- R
- MATLAB
- SAS (Statistical Analysis System)
- Scala
- JAVA
- C++
- Hadoop

Expert System

- In artificial intelligence, an expert system is a computer system that **emulates the decision-making ability** of a human expert.

- Expert systems are designed to solve complex problems by reasoning through bodies of knowledge, represented mainly as if–then rules rather than through conventional procedural code.

- The first expert systems were created in the 1950s and then production increases in the 1980s.

- Expert systems were among the first truly successful forms of artificial intelligence. For Example
  - MYCIN: To identify various bacteria that can cause severe infections and can also recommend drugs based on the person's weight.
  - CaDet: It is a clinical support system that could identify cancer in its early stages in patients.

How do humans make decisions?

Human decision making

 skin rash ✓

 fever ✗

 headache ✓

 cold cough ✗

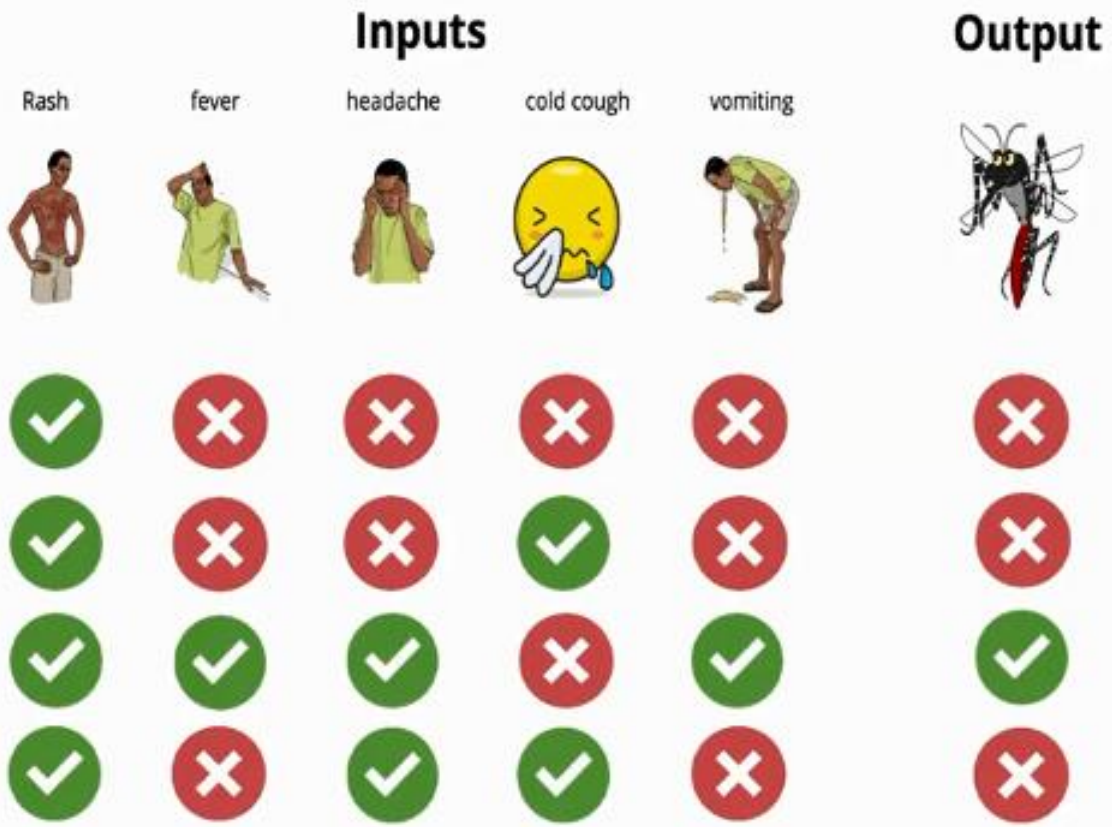
 vomiting ✓

 ✗

Dengue

How do humans make decisions from past experiences?

Human decision making



How do humans make decisions from past experiences?

Human decision making

inputs					output
Rash	fever	headache	cold cough	vomiting	
					
1	0	0	0	0	0
1	0	0	1	0	0
1	1	1	0	1	1
1	0	1	1	0	0

✗	0
✓	1

Is this applicable in multiple domains ?

Human decision making

Inputs | **Output**

Pitching in line	Impact	Height	no ball	shot attempted	LBW out
					
✓	✓	✗	✓	✗	✗
✓	✓	✓	✗	✗	✓
✓	✓	✓	✗	✓	✓
✓	✗	✓	✓	✗	✗

Is this applicable in multiple domains ?

Human decision making

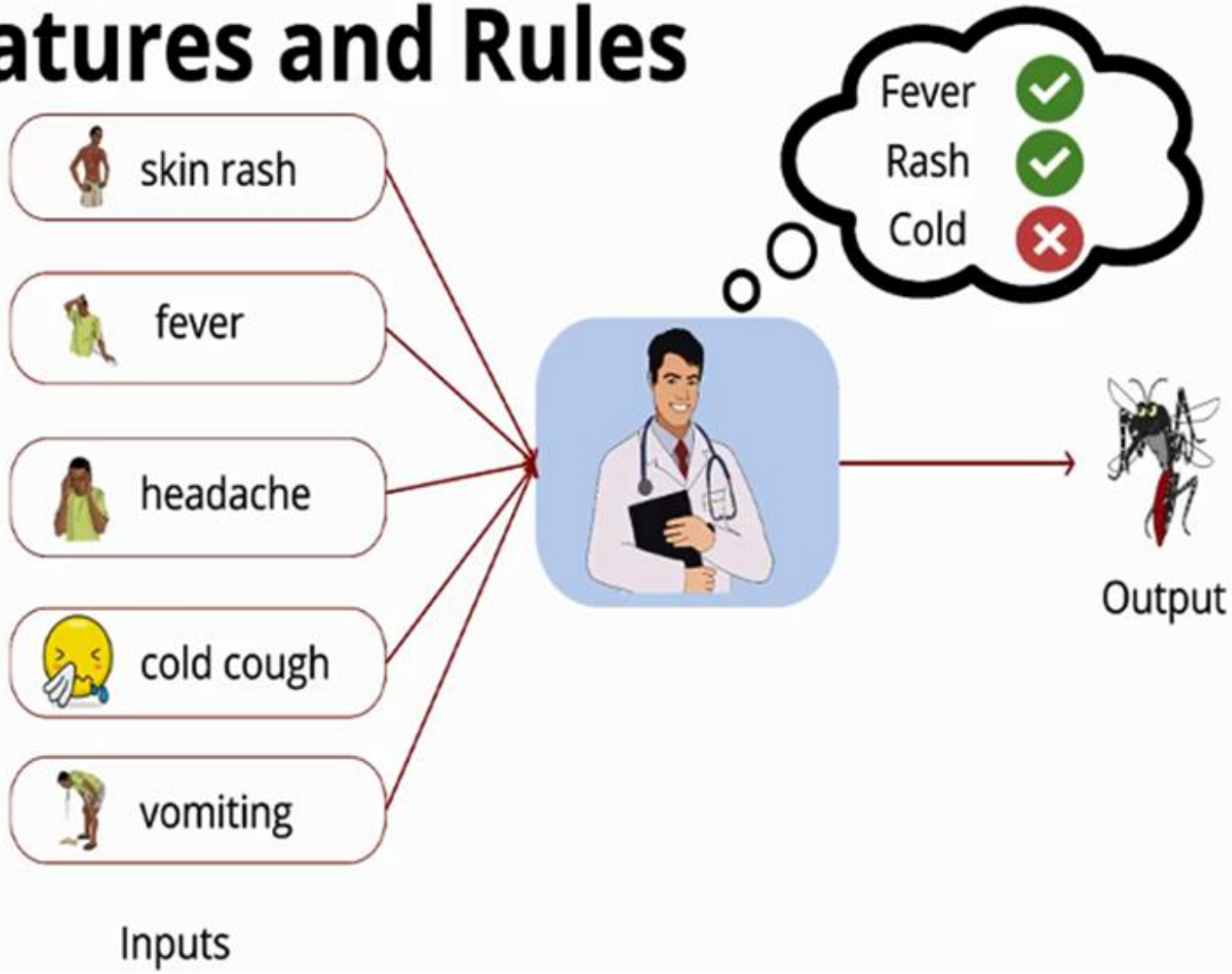
Inputs					Output
Pitching in line	Impact	Height	no ball	shot attempted	LBW out
					

1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	0	1	1
1	0	1	1	0	0

✗	0
✓	1

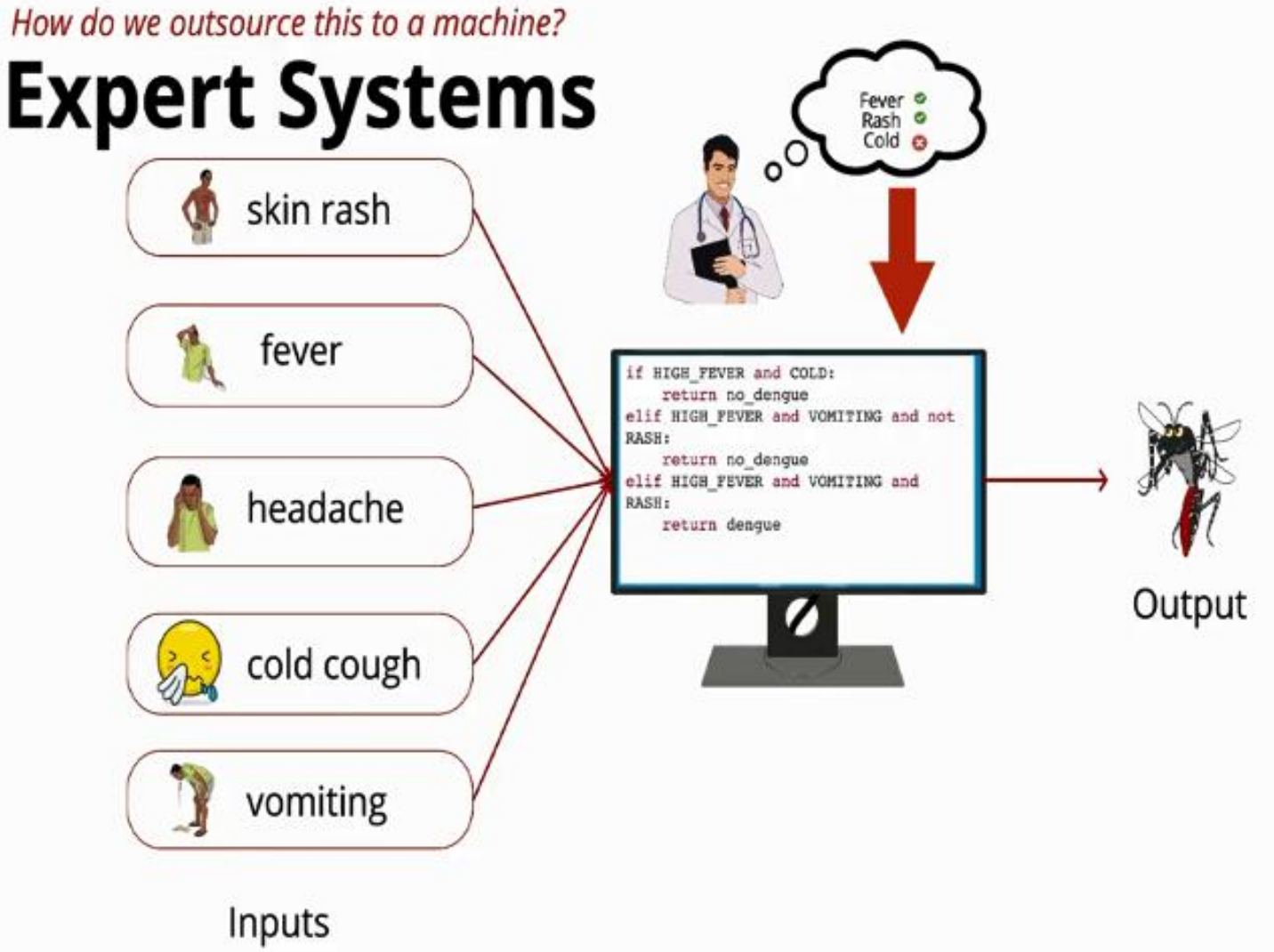
What is the semantics of decision making?

Features and Rules



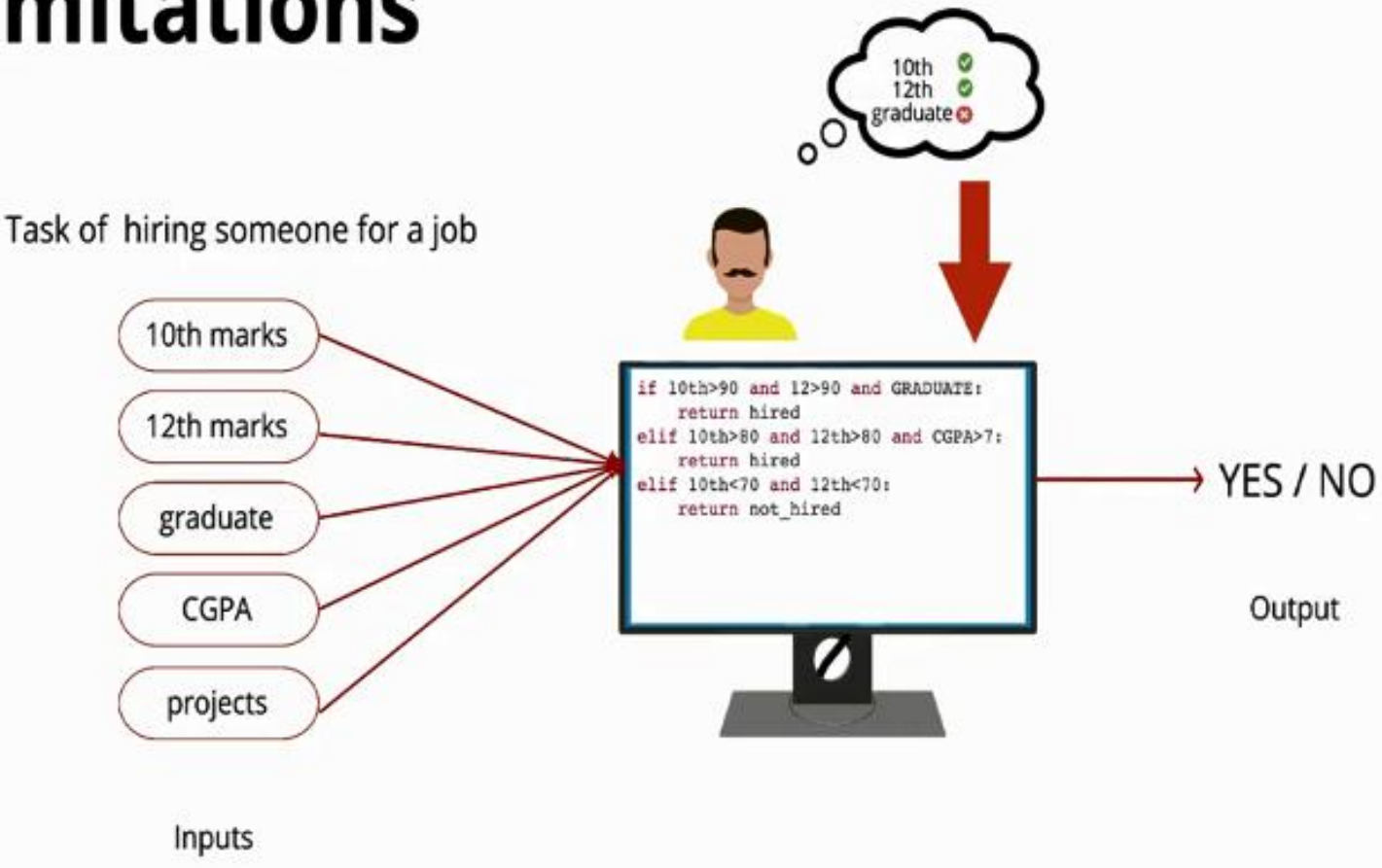
How do we outsource this to a machine?

Expert Systems



Do we need to look beyond expert systems?

Limitations

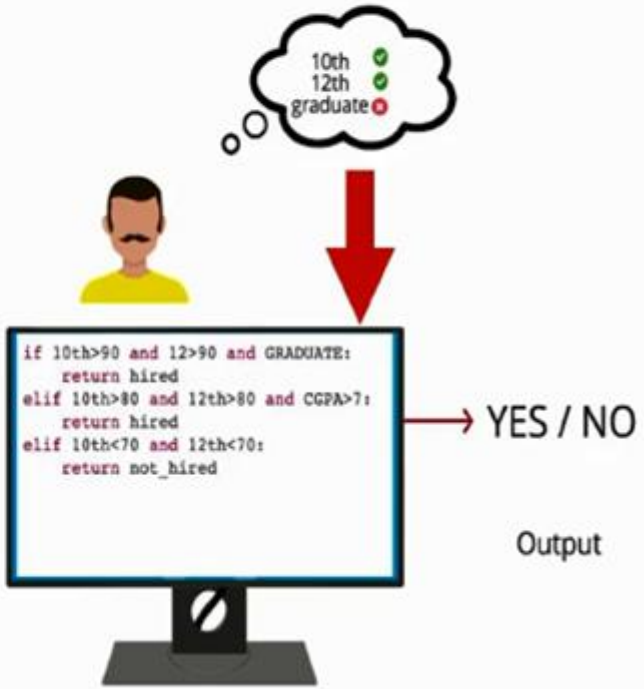


Do we need to look beyond expert systems?

Limitations

Lots of data to make sense from

10th marks	12th marks	graduate	CGPA	projects	.....	awards
92	82	yes	9.2	3	.....	2
83	75.2	yes	7.2	1	.....	0
75	70	no	6.4	0	.....	1
96	95	yes	9.5	5	.....	4
90	89	yes	8.8	2	.....	1
78	82	yes	7.6	0	.....	0
86	88	yes	8.4	1	.....	1

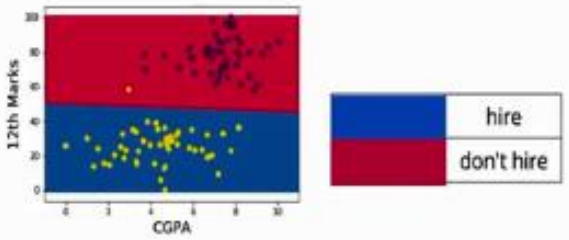


1) Huge amount of data
2) Some time difficult to code express honesty, confidence, facial expression etc. that can be achieve by interaction
3) Some time rule can be unknown

How to move from writing rules to learning rules?

Say Hi to Machine Learning

$$\hat{y} = f(x_1, x_2)$$



10th marks	12th marks	graduate	CGPA	projects	.....	awards
92	82	yes	9.2	3	.....	2
83	75.2	yes	7.2	1	.....	0
75	70	no	6.4	0	.....	1
96	95	yes	9.5	5	.....	4
90	89	yes	8.8	2	.....	1
78	82	yes	7.6	0	.....	0
86	88	yes	8.4	1	.....	1

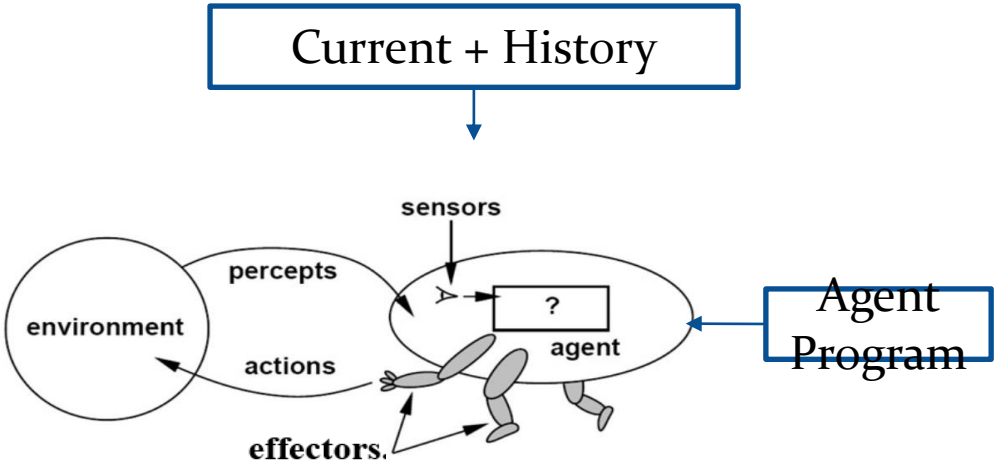
→ YES / NO

Output

A shift from rule-based approach to a data-driven

Agents

- An AI system is composed of an **agent and its environment**. The agents act in their environment. The environment may contain other agents.
- An **agent** is anything that can perceive its environment through **sensors** and acts upon that environment through **effectors.**



- **Sensor:** Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.
- **Actuators:** Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, etc.
- **Effectors:** Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.

Examples of Agent

- A human agent has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.

- A robotic agent replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.

Goals of Agent

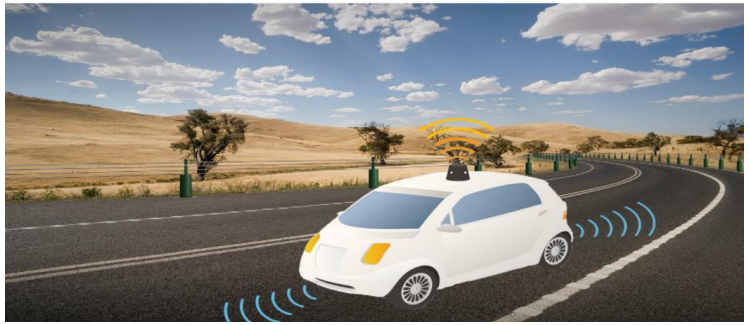
- High Performance

- Optimized Result

- Rational Action (Right or accurate)

PEAS Representation

- PEAS is a type of model on which an AI agent works upon.
  - **P:** Performance measure
  - **E:** Environment
  - **A:** Actuators
  - **S:** Sensors



Let's suppose a self-driving car then PEAS representation will be:
Performance: Safety, time, legal drive, comfort
Environment: Roads, other vehicles, road signs
Actuators: Steering, accelerator, break, signal, horn
Sensors: Camera, GPS, speedometer, odometer, accelerometer, sonar.

Types of Environment

Fully Observable vs Partially Observable Environment

- If an agent sensor can sense or access the **complete state** of an environment at each point of time then it is a fully observable environment, else it is partially observable.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.
- An agent with no sensors in all environments then such an environment is called as unobservable.

Ex.

- Chess – the board is fully observable, and so are the opponent's moves.
- Driving – the environment is partially observable because what's around the corner is not known.

Deterministic vs Stochastic:

- If an **agent's current state and selected action can** completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.

Single-agent vs Multi-agent

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.
- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.

Static vs Dynamic:

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment.
- Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action.
- However for dynamic environment, agents need to keep looking at the world at each action.
- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

Discrete vs Continuous:

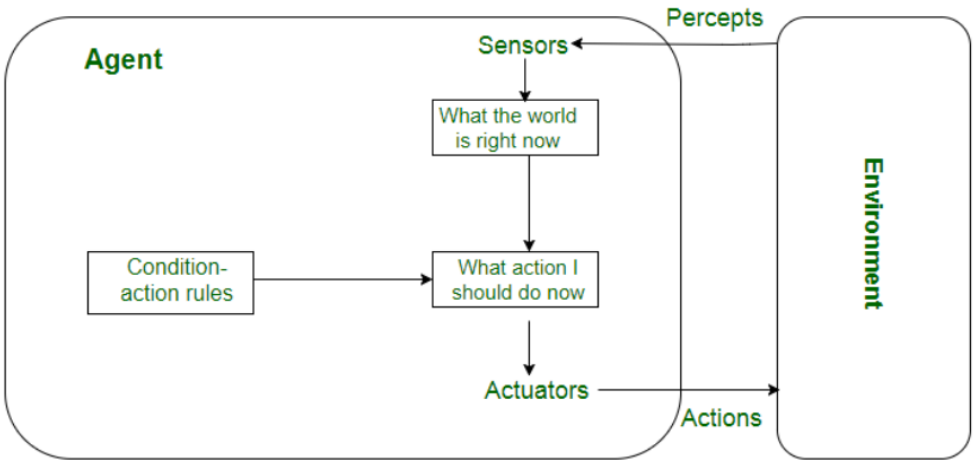
- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess game comes under discrete environment as there is a finite number of moves that can be performed.

Types of Agents

- Simple Reflex Agents
- Model-Based Reflex Agents
- Goal-Based Agents
- Utility-Based Agents
- Learning Agent

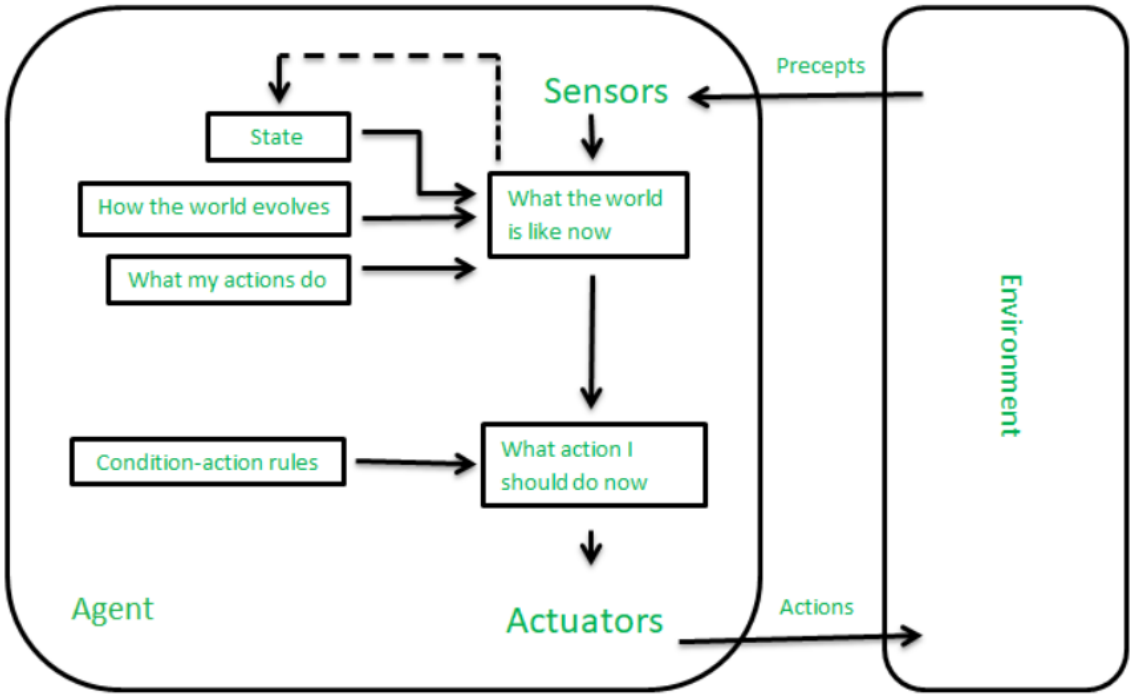
Simple Reflex Agents

- Simple reflex agents ignore the rest of the percept history and act only on the basis of the **current percept**.
- The agent function is based on the **condition-action rule**.
- If the condition is true, then the action is taken, else not.
- This agent function only succeeds when the environment is **fully observable** (i.e. not suitable for **partial observable**).



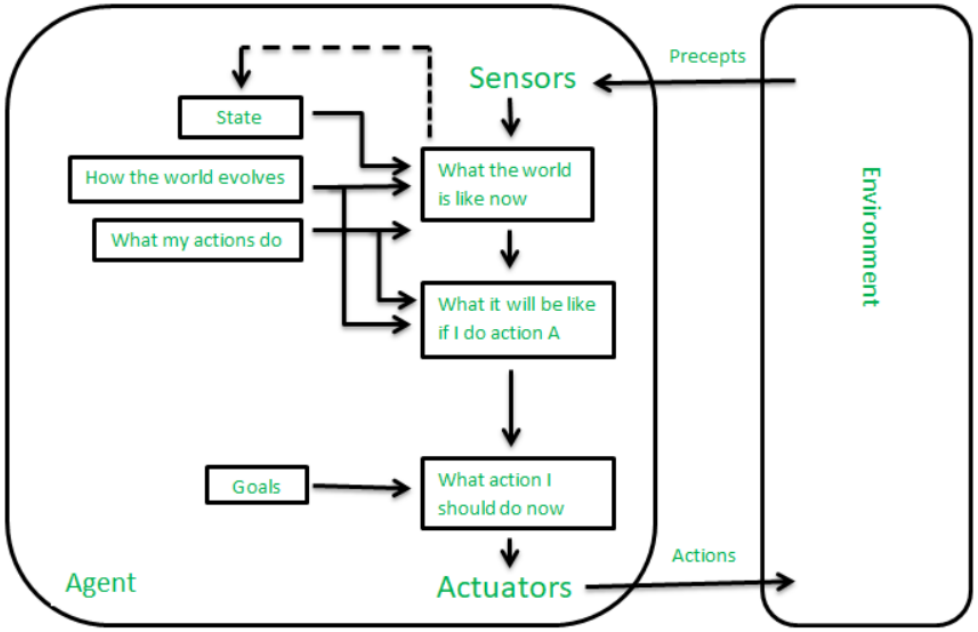
Model-based Reflex Agents

- It works by finding a rule whose condition matches the current situation.
- A model-based agent can handle **partially observable environments**.
- The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.



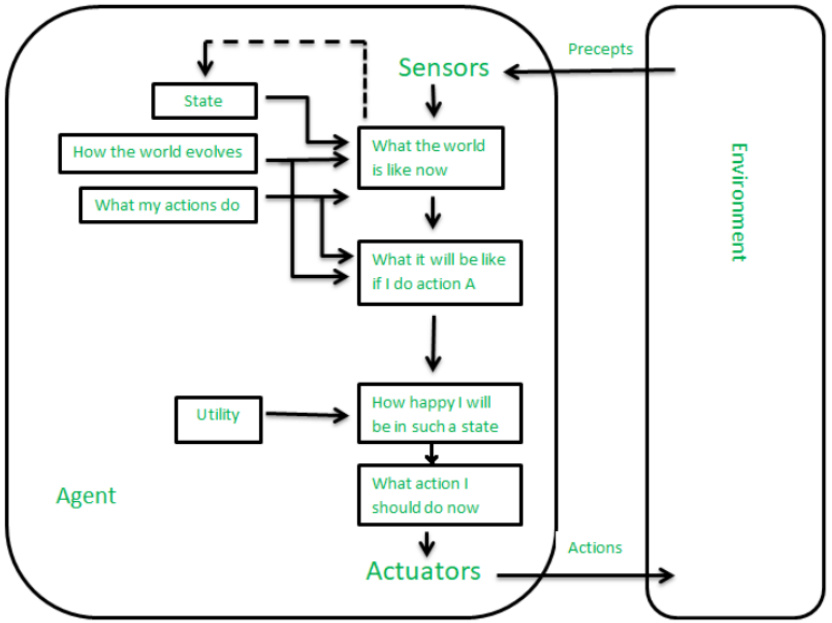
Goal-based Agents

- These kinds of agents take decisions based on how far they are currently from their goal(description of desirable situations).
- Their every action is intended to reduce its distance from the goal.
- They usually require **search and planning**. The goal-based agent's behavior can easily be changed.



Utility-based Agents

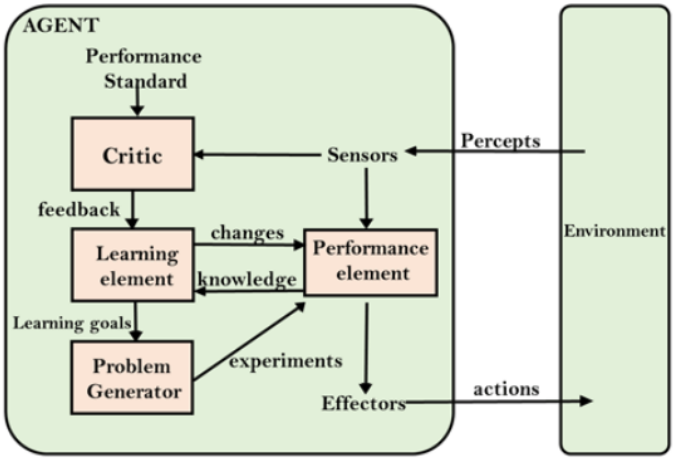
- The agents which are developed having their end uses as building blocks are called utility-based agents.
  - When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used. They choose actions based on a preference (utility) for each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a destination.
- Agent happiness should be taken into consideration. Utility describes how "**happy**" the agent is.



Learning Agent

- A learning agent in AI is the type of agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act and adapt automatically through learning. A learning agent has mainly four conceptual components, which are:

1. **Learning element:** It is responsible for making improvements by learning from the environment
2. **Critic:** The learning element takes feedback from critics which describes how well the agent is doing with respect to a fixed performance standard.
3. **Performance element:** It is responsible for selecting external action
4. **Problem Generator:** This component is responsible for suggesting actions that will lead to new and informative experiences.



Rational and Intelligent Agent

❖ Intelligent Agent

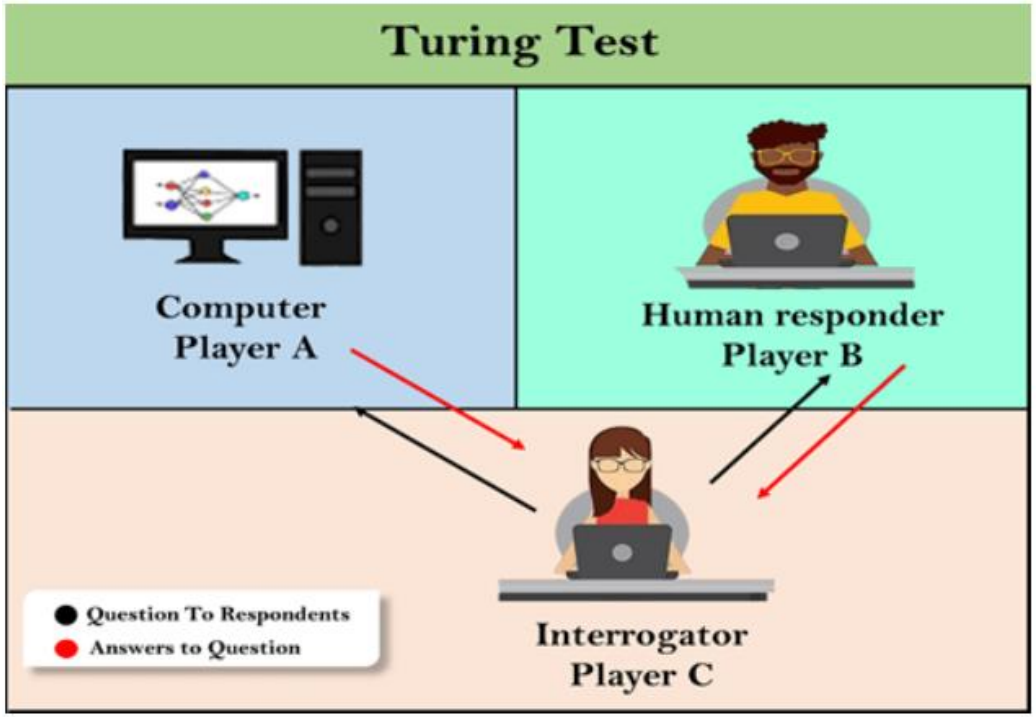
- An intelligent agent is a goal-directed agent. It perceives its environment through its sensors using the observations and built-in knowledge, acts upon the environment through its actuators.

❖ Rational Agent

- Rationality is nothing but status of being **reasonable, sensibl**e, and having **good sense of judgment**.
- Rationality is concerned with expected actions and results depending upon what the agent has perceived.
- A rational agent always performs right action. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).

Turing Test in AI

- In 1950, Alan Turing introduced a test to check whether a machine can think like a human or not, this test is known as the Turing Test. I
- n this test, Turing proposed that the computer can be said to be an intelligent if it can mimic human response under specific conditions.



General Problem Solving Components

- Every problem should be properly formulated in artificial intelligence. Problem formulation is very important before applying any search algorithm.

- In AI one must identify components of problems, which are:-

- Problem Statement
  - Definition
  - Limitation or Constraints or Restrictions
- Problem Solution
- Solution Space
- Operators

❖ Definition of Problem

- The information about what is to be done? Why it is important to build AI system? What will be the advantages of proposed system? For example "**I want to predict the price of house using AI system**".

❖ Problem Limitation

- There always some limitations while solving problems. All these limitations or constraints must be fulfill while creating system. For example "**I have only few features, some records are missing. System must be 90% accurate otherwise it will be useless**".

❖ Goal or Solution

- What is expected from system? The Goal state or final state or the solution of problem is defined here. This will help us to proposed appropriate solution for problem. For example "**we can use some machine learning technique to solve this problem".**

❖ Solution Space

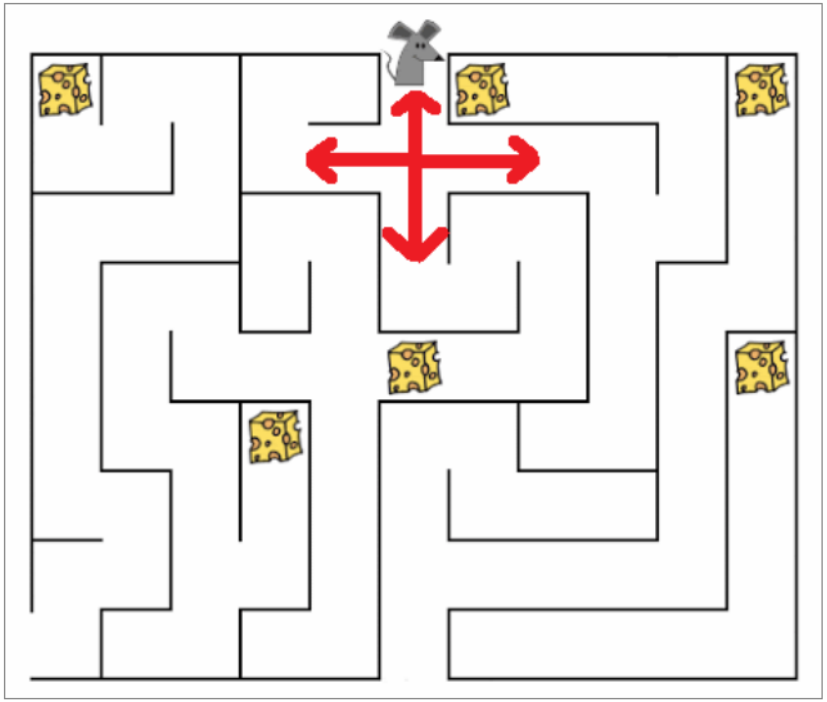
- Problem can be solve in many ways. Some solution will be efficient than others. Some will consume less resources, some will be simple etc. There are always alternatives exists. Many possible ways with which we can solve problem is known as Solution Space. For example "**price of house can be predict using many machine learning algorithms**".

❖ Operators

- **Operators are the actions taken during solving problem**. Complete problem is solved using tiny steps or actions and all these consecutive actions leads to solution of problem.

Mouse Path Problem

- **Problem Statement**
  - **Problem Definition:** Mouse is hungry, mouse is in a puzzle where there are some cheese. Mouse will only be satisfied if mouse eat cheese
  - **Problem Limitation:** Some paths are close i-e dead end, mouse can only travel through open paths
- **Problem Solution**: Reach location where is cheese and eat minimum one cheese. There are possible solutions (cheese pieces)
- **Solution Space:** To reach cheese there are multiple paths possible
- **Operators**: Mouse can move in four possible directions, these directions are operators or actions which are UP, DOWN, LEFT and RIGHT

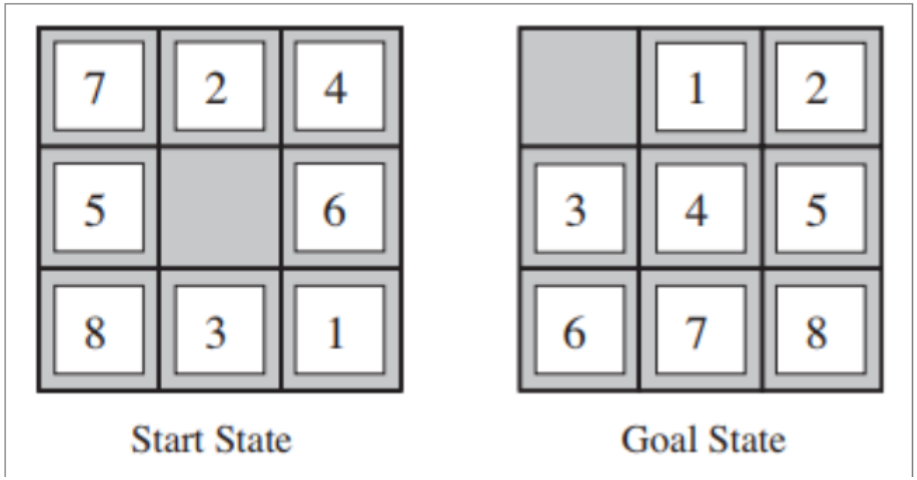


Water Jug Problem

- **Problem Statement**
  - **Problem Definition:** You have to measure 4 liter (L) water by using three buckets 8L, 5L and 3L.
  - **Problem Limitation:** You can only use these (8L, 5L and 3L) buckets
- **Problem Solution**: Measure exactly 4L water
- **Solution Space:** There are multiple ways doing this.
- **Operators**: Possible actions are fill water in any bucket and remove water from any bucket.

8 Puzzle or Slide Puzzle

- **States**: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state**: Any random shuffled state can be designated as initial state
- **Actions**:
  - Slide Left
  - Slide Right
  - Slide Up
  - Slide Down
- **Transition model**: Given a **state** and **action**, this returns the resulting state
- **Goal test**: This checks whether the state matches the goal
- **Path cost**: Each step costs 1



Search Algorithm Terminologies

- In Artificial Intelligence, Search techniques are universal problem-solving methods.

- **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
  - **Search Space:** Search space represents a set of possible solutions, which a system may have.
  - **Start State:** It is a state from where agent begins **the search**.
  - **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the **initial state**.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.

Properties of Search Algorithms

- **Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

- **Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

- **Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.

- **Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

- **Systematicity**: Does it visited each node at most once.

Types of search algorithms

❖ Uninformed/Blind Search

- The uninformed search does not contain any domain knowledge such as closeness, the location of the goal.
- It operates in a brute-force way as it only includes information about how to traverse the tree and how to identify leaf and goal nodes.

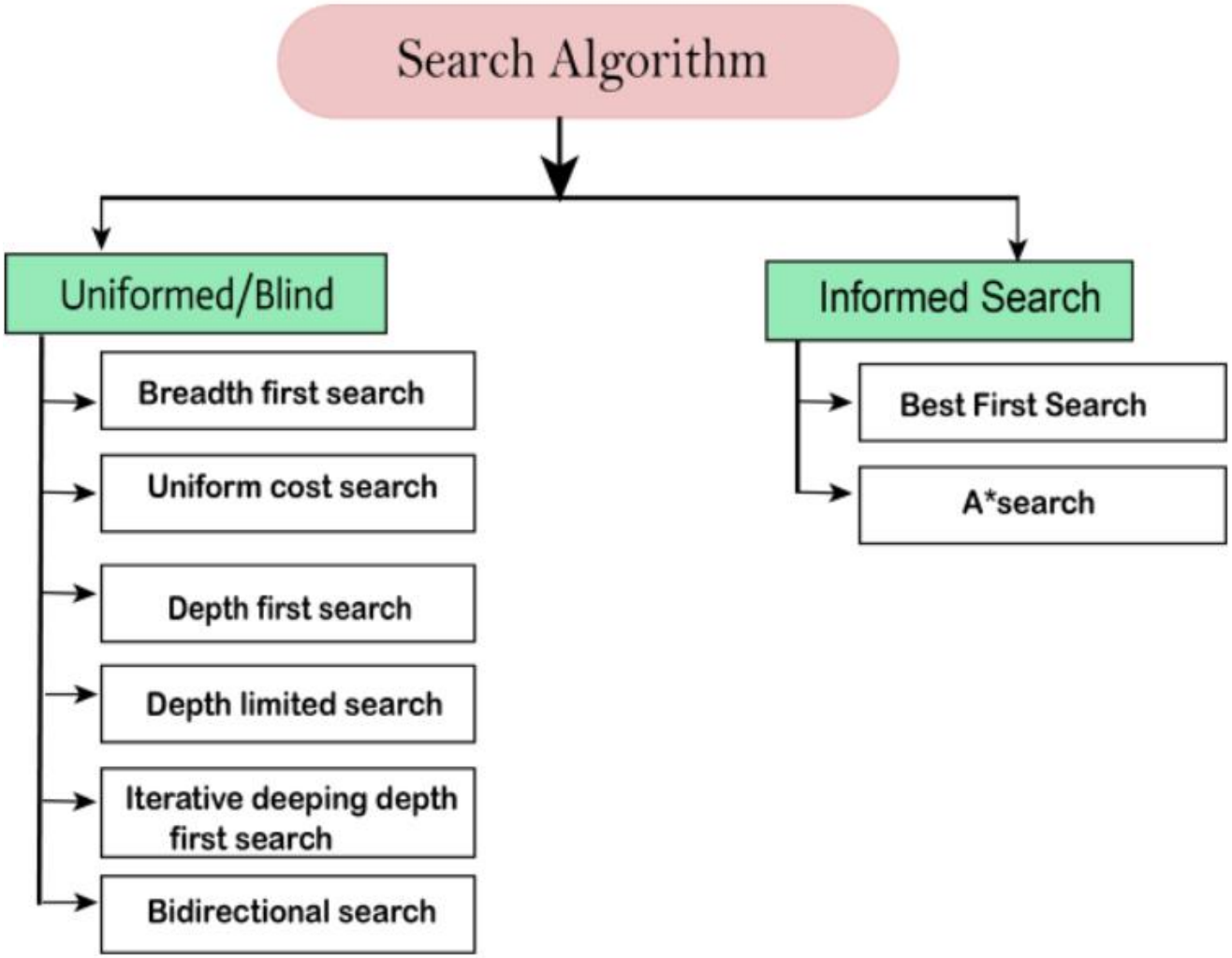
❖ Informed Search

- Informed search algorithms use domain knowledge. In an informed search, problem information is available which can guide the search.
- Informed search strategies can find a solution more efficiently than an uninformed search strategy.

Difference between Informed and Uninformed Search

Informed Search	Uninformed Search
It uses knowledge for the searching process.	It doesn't use knowledge for searching process.
It finds solution more quickly.	It finds solution slow as compared to informed search.
Cost is low.	Cost is high.
It consumes less time.	It consumes moderate time.
It provides the direction regarding the solution.	No suggestion is given regarding the solution in it.
Less complexity (time, space)	More complexity (time, space)
Greedy Search, A* Search, Graph Search	Depth First Search, Breadth First Search

Types of search algorithms



Node Data Structure

- A node used in the search algorithm is a data structure which contains the following:
  1. A state description
  2. A pointer to the parent of the node
  3. Depth of the node
  4. The operator that generated this node
  5. Cost of this path (sum of operator costs) from the start state

The Basic Search algorithm

```
Let L be a list containing the initial state (L= the fringe)
Loop
        if L is empty return failure
        Node ← select (L)
         if Node is a goal
                  then return Node
                     (the path from initial state to Node)
        else generate all successors of Node, and
             merge the newly generated states into L
End Loop
```

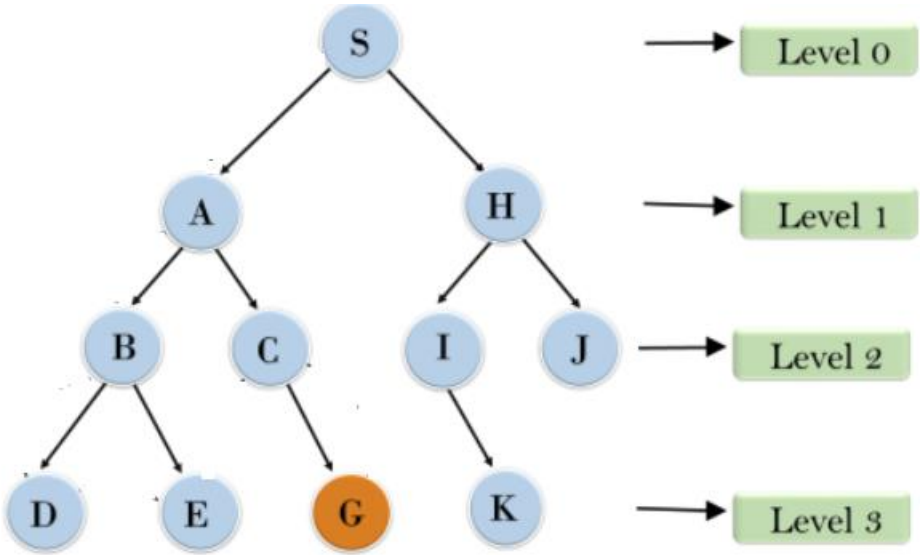
For Tree

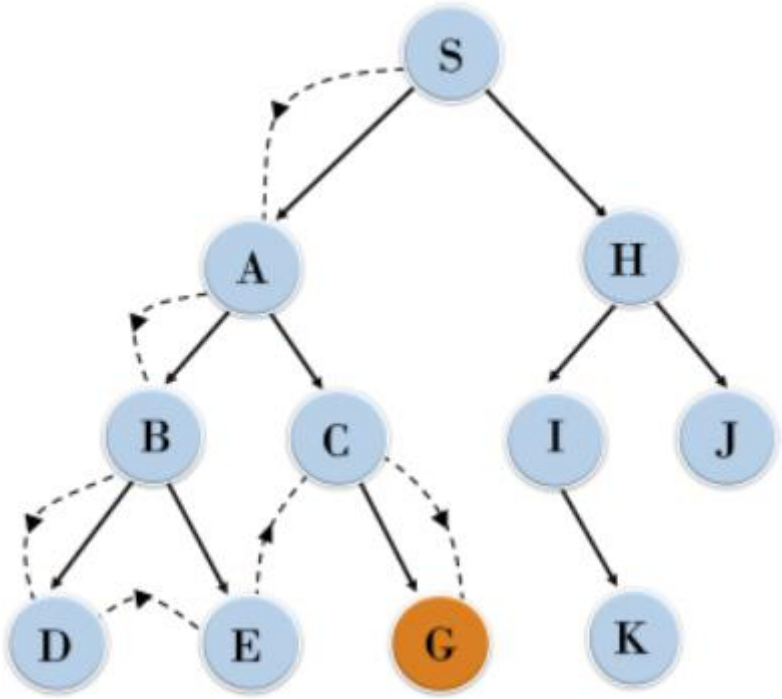
```
Graph search algorithm
Let fringe be a list containing the initial state
Let closed be initially empty
Loop
        if fringe is empty return failure
        Node ← remove-first (fringe)
         if Node is a goal
            then return the path from initial state to Node
        else put Node in closed
                generate all successors of Node S
                for all nodes m in S
                    if m is not in fringe or closed
                        merge m into fringe
End Loop
```

For Graph

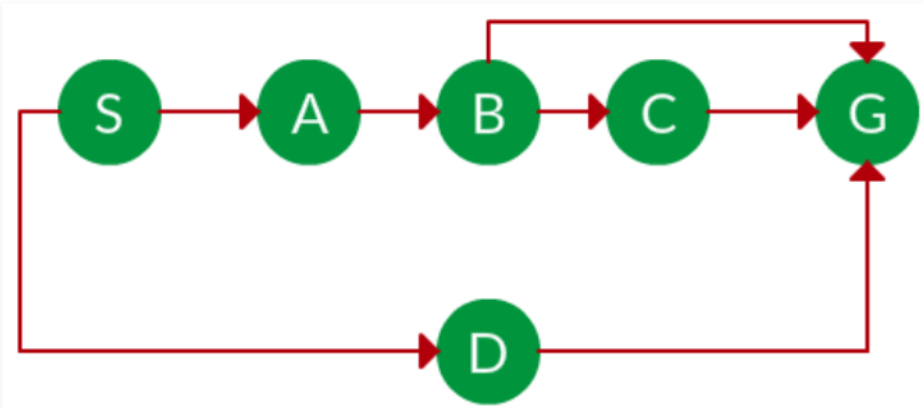
1. Depth-first Search

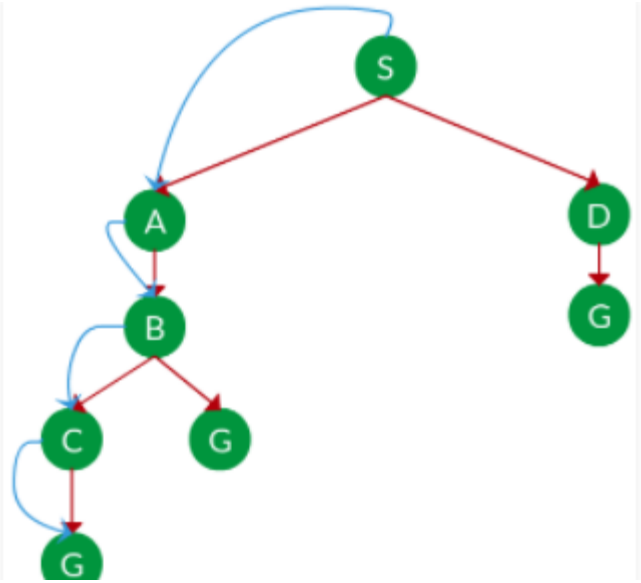
- It starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a **stack** data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.


S
A
H
B
C
I
J
D
E
G
K
Level 0
Level 1
Level 2
Level 3


S
A
H
B
C
I
J
D
E
G
K

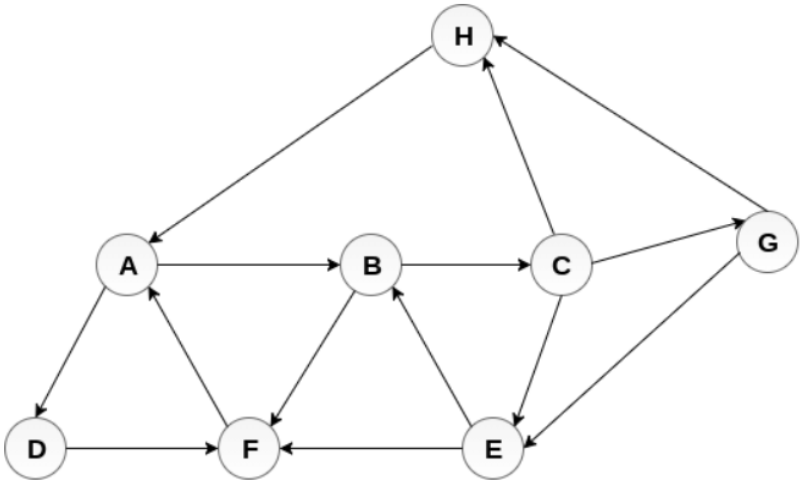
The equivalent search tree the graph





S -> A -> B -> C -> G

Calculate the order to print all the nodes of the graph starting from node H to E



H → A → D → F → B → C → G → E

Algorithm

Depth First Search

Let *fringe* be a list containing the initial state
Loop
    if *fringe* is empty return failure
    Node ← remove-first (*fringe*)
     if Node is a goal
        then return the path from initial state to Node
    else generate all successors of Node, and
        merge the newly generated nodes into *fringe*
        add generated nodes to the front of *fringe*
End Loop

Applications of DFS

1) We can detect cycles in a graph using DFS. If we get one back-edge during BFS, then there must be one cycle.

DFS not BFS

2) Using DFS we can find path between two given vertices u and v.

3) We can perform topological sorting is used to scheduling jobs from given dependencies among jobs. Topological sorting can be done using DFS algorithm.

4) Using DFS, we can find strongly connected components of a graph. If there is a path from each vertex to every other vertex, that is strongly connected.

Properties of DFS

Completeness: Not complete. It is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

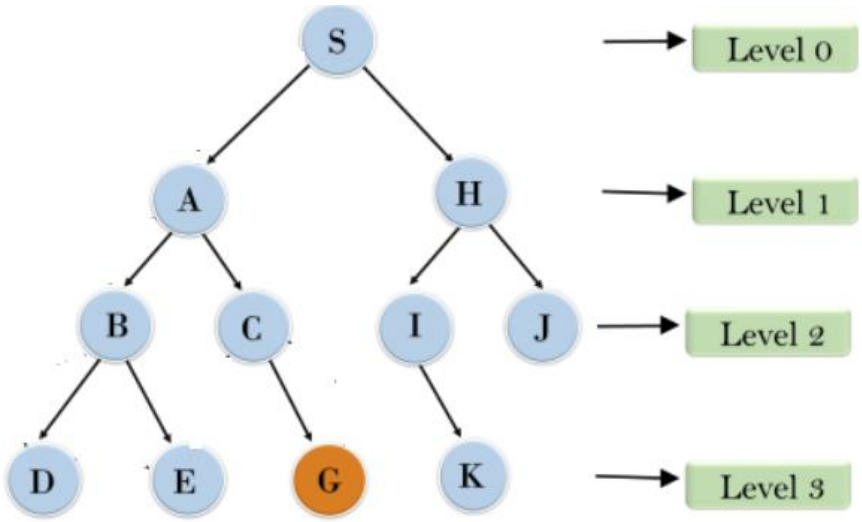
$T(n)= 1+ b^2+ b^3 +.....+ b^m=O(b^m)$

Where, m= maximum depth of any node and b is the branch factor)

Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

Completeness
Time Complexity
Space Complexity
Optimal



Advantages and Disadvantages of DFS

❖Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

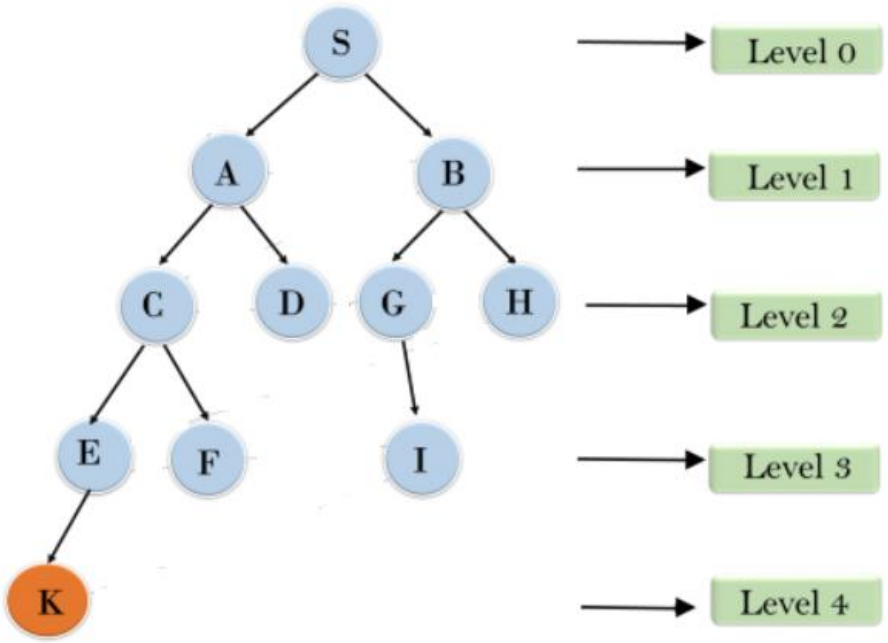
❖Disadvantage:

- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

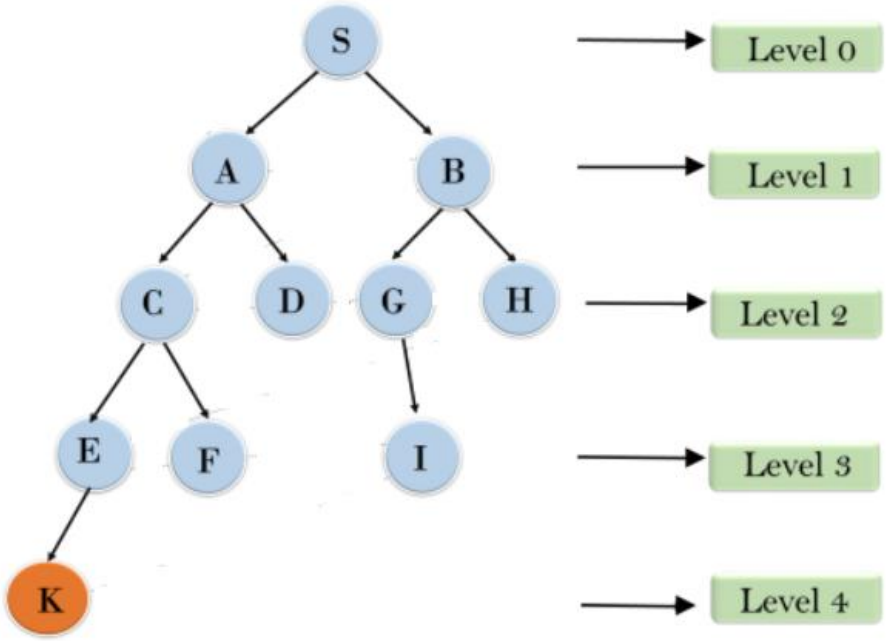
2. Breadth-first Search

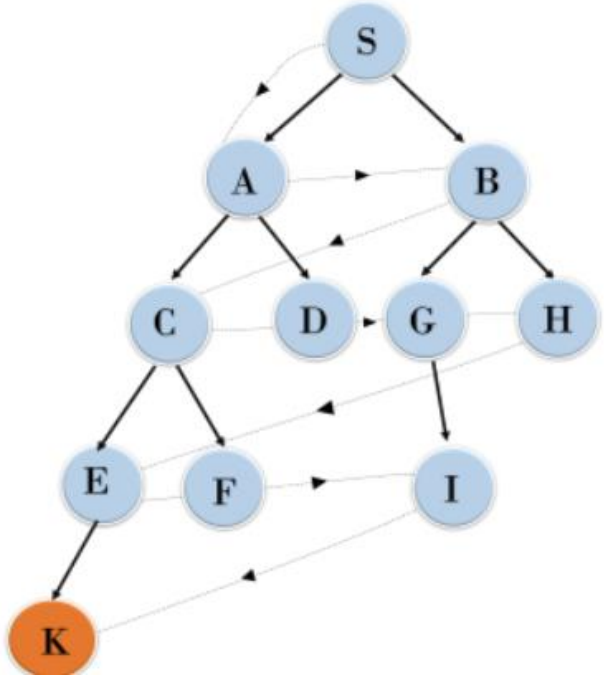
- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the **root node** of the tree and **expands all successor node at the current level** before moving to nodes of next level.
- Breadth-first search implemented using **FIFO queue** data structure.

BFS algorithm from the root node S to goal node K



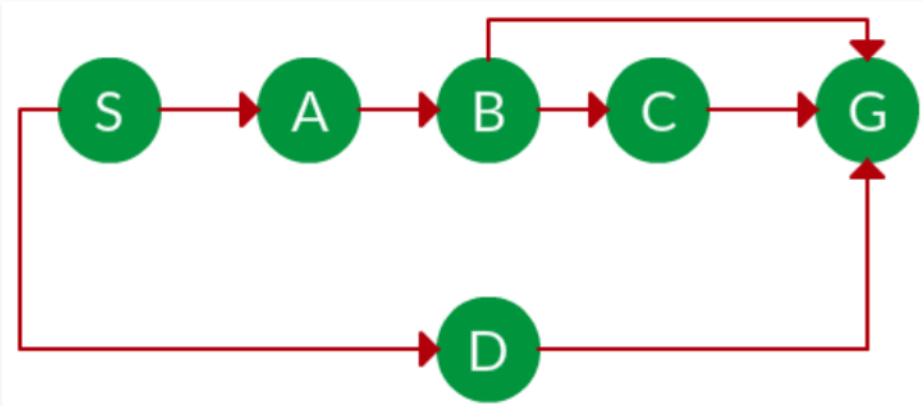
BFS algorithm from the root node S to goal node K

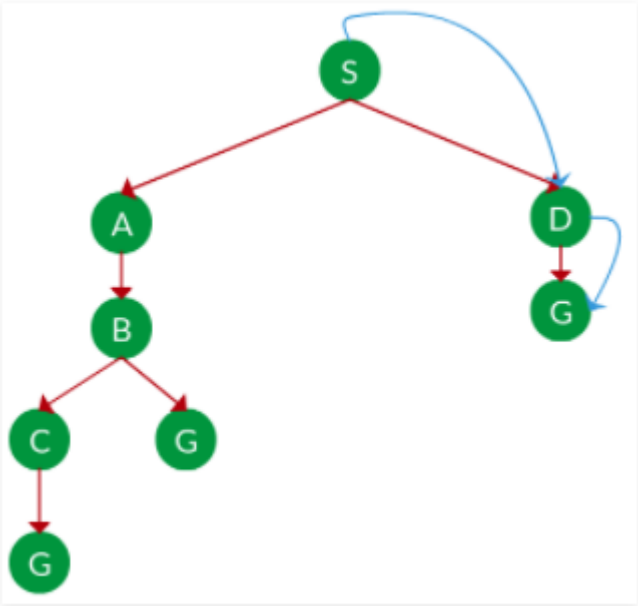




S---> A--->B---->C--->D---->G--->H--->E---->F---->I---->K

The equivalent search tree for the graph





Path: S -> D -> A -> G

Algorithm for Tree

Breadth first search
Let *fringe* be a list containing the initial state
Loop
    if *fringe* is empty return failure
    Node ← remove-first (fringe)
     if Node is a goal
        then return the path from initial state to Node
    else generate all successors of Node, and
        (merge the newly generated nodes into *fringe*)
        add generated nodes to the back of *fringe*
End Loop

Algorithm for Graph

Graph search algorithm
Let fringe be a list containing the initial state
Let closed be initially empty
Loop
    if fringe is empty return failure
    Node ← remove-first (fringe)
    if Node is a goal
        then return the path from initial state to Node
    else put Node in closed
        generate all successors of Node S
        for all nodes m in S
            if m is not in fringe or closed
                merge m into fringe
End Loop

Applications of BFS

1) Search **engine crawlers** are used BFS to build index. Starting from source page, it finds all links in it to get new pages
2) Using GPS navigation system BFS is used to find **neighboring places**.
3) In networking, when we want to **broadcast some packets**, we use the BFS algorithm.
4) Path finding algorithm is based on BFS or DFS.

Properties of BFS

Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

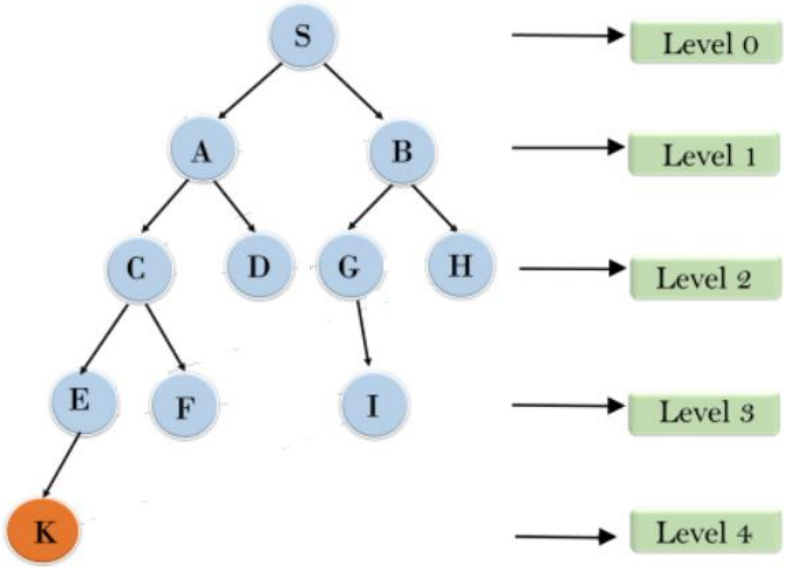
$T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if **path cost is a non-decreasing** function of the depth of the node.

Completeness
Time Complexity
Space Complexity
Optimal



Advantages and Disadvantages of BFS

❖Advantages:

- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

❖Disadvantages:

- It requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

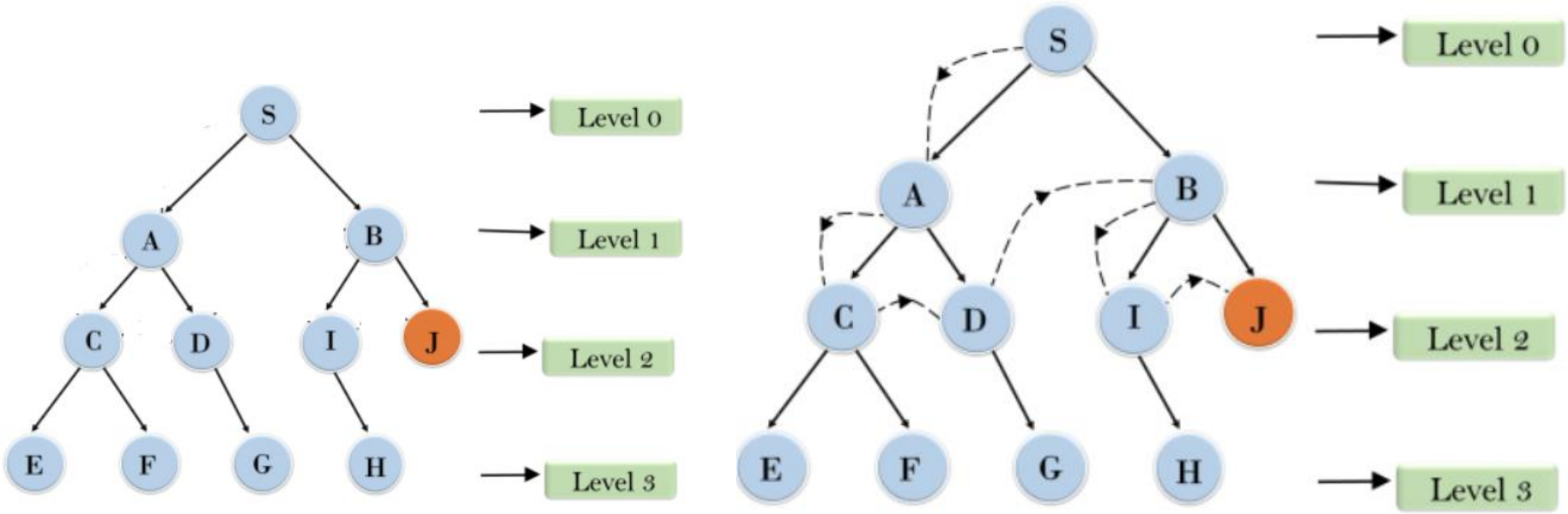
3. Depth-Limited Search Algorithm

- A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

- Depth-limited search can be terminated with two Conditions of failure:

  1) Standard failure value: It indicates that problem does not have any solution.

  2) Cutoff failure value: It defines no solution for the problem within a given depth limit.



Advantages:

- Depth-limited search is Memory efficient.

Disadvantages:

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

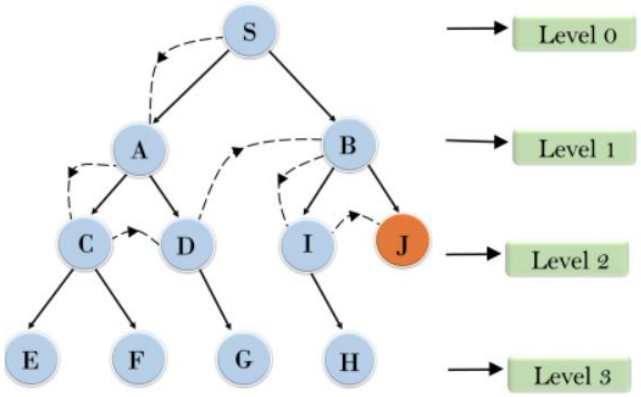
Algorithm

```
Depth limited search (limit)
Let fringe be a list containing the initial state
Loop
        if fringe is empty return failure
        Node ← remove-first (fringe)
         if Node is a goal
            then return the path from initial state to Node
       else if depth of Node = limit return cutoff
       else add generated nodes to the front of fringe
End Loop
```

Properties of Depth-Limited Search

- **Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.
- **Time Complexity:** Time complexity of DLS algorithm is **O(b^{ℓ})**.
- **Space Complexity:** Space complexity of DLS algorithm is O**($b \times \ell$)**.
- **Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if $\ell > d$.

Completeness
Time Complexity
Space Complexity
Optimal



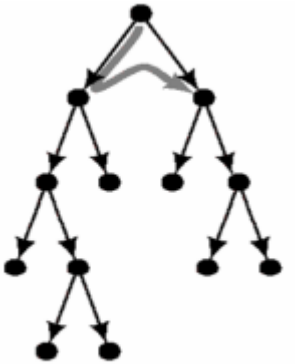
4. Depth-First Iterative Deepening (DFID)

- First do DFS to depth 0 (i.e., treat start node as having no successors), then, if no solution found, do DFS to depth 1, etc.

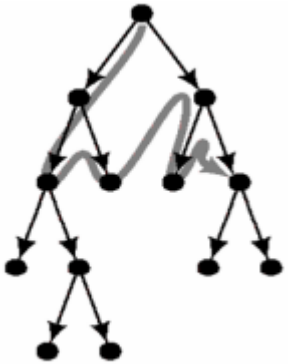
DFID
until solution found do *DFS with depth cutoff c* *c = c+1*

- Advantage
  - ✓ Linear memory requirements of depth-first search
  - ✓ Guarantee for goal node of minimal depth

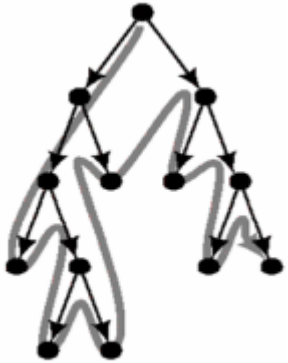
Procedure



Depth bound = 1



Depth bound = 2



Depth bound = 4

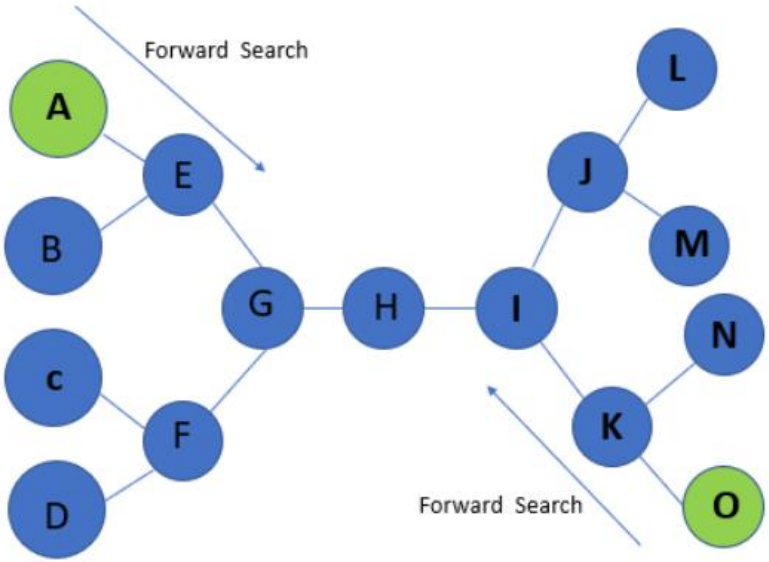
Depth bound = 3

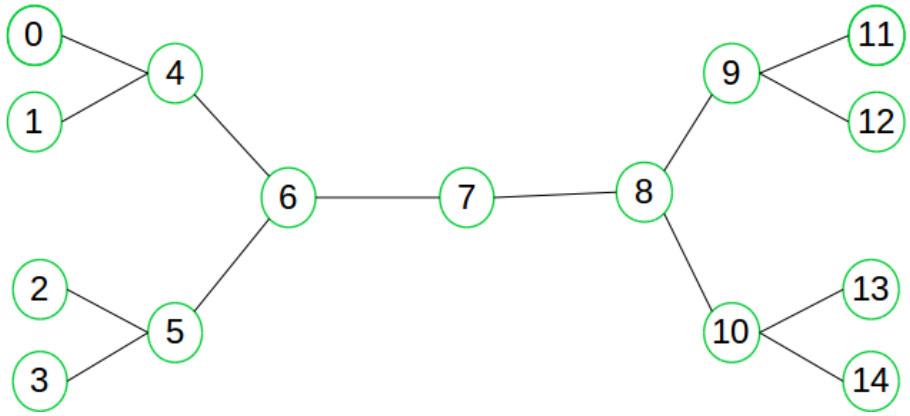
Property

- **Complete:** Yes
- **Optimal** if all operators have the same cost. Otherwise, not optimal but guarantees finding solution of shortest length (like BFS).
- **Time compl**exity is a little worse than BFS or DFS because nodes near the top of the search tree are generated multiple times, but because almost all of the nodes are near the bottom of a tree, the worst case time complexity is still exponential, O(b^d)
- **Linear space complexity**, O(bd), like DFS

5. Bidirectional Search Algorithm

- In normal graph search using BFS/DFS we begin our search in one direction usually from source vertex toward the goal vertex, but what if we start search from both direction simultaneously.
- Bidirectional search is a graph search algorithm which find smallest path from source to goal vertex in a **directed graph**. It runs two simultaneous search –
  1) Forward search from source/initial vertex toward goal vertex
  2) Backward search from goal/target vertex toward source vertex
- Bidirectional search replaces single search graph(which is likely to grow exponentially) with two smaller sub graphs – one starting from initial vertex and other starting from goal vertex. The search terminates when two graphs intersect.





- Suppose we want to find if there exists a path from vertex 0 to vertex 14. Here we can execute two searches, one from vertex 0 and other from vertex 14.
- When both forward and backward search meet at vertex 7, we know that we have found a path from node 0 to 14 and search can be terminated now.

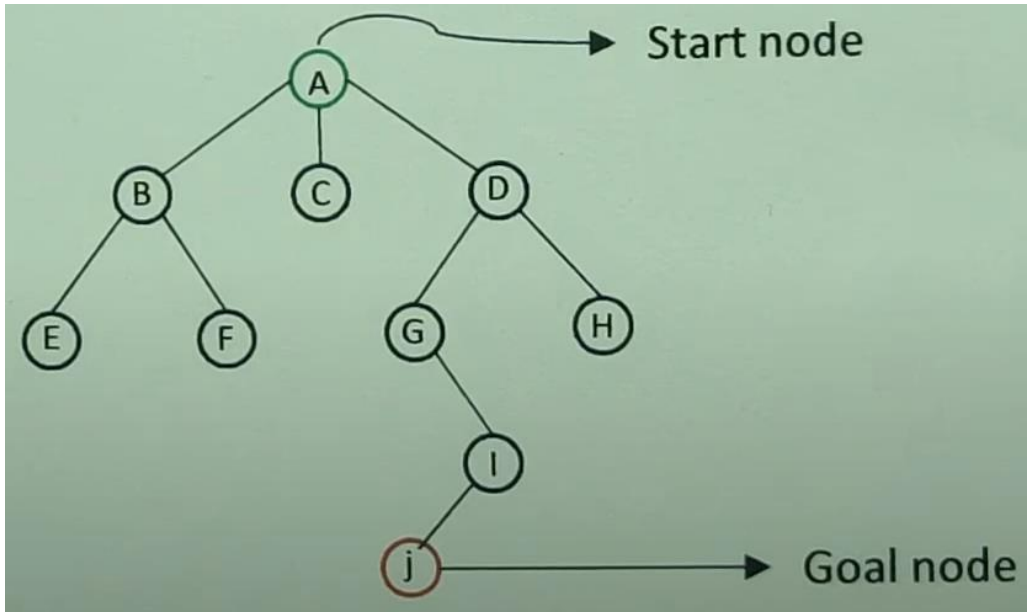
When to use bidirectional approach?

1. Both initial and goal states are **unique** and completely defined.
2. The branching factor is exactly the same in both directions.

Completeness
Time Complexity
Space Complexity
Optimal

Performance measures

- Completeness : Bidirectional search is complete if BFS is used in both searches.
- Optimality : It is optimal if BFS is used for search and paths have uniform cost.
- Space Complexity : Space complexity is same as BFS and DFS.
- Total time complexity would be $O(b^{d/2} + b^{d/2})$ which is far less than $O(b^d)$.

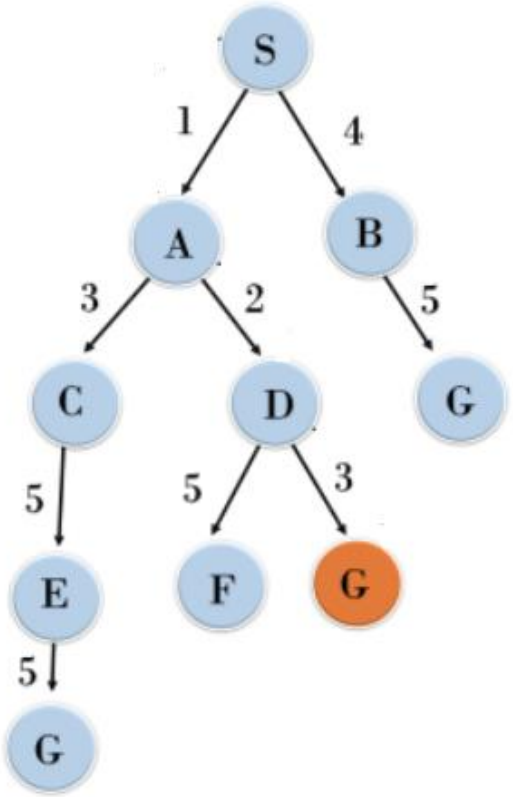


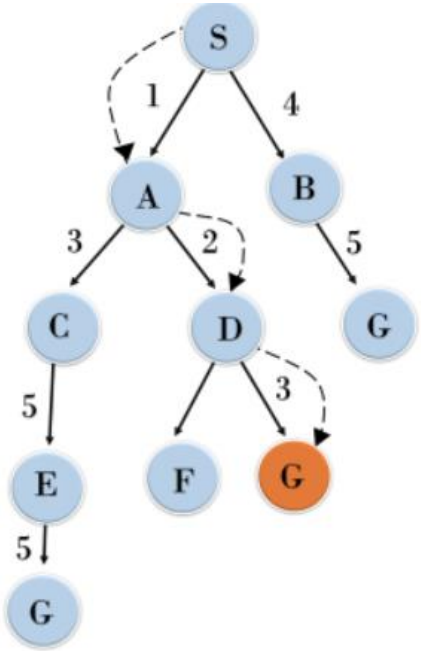
A -> B -> C -> D

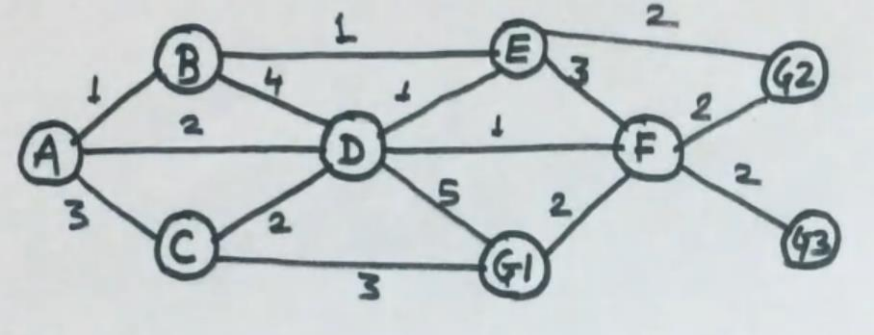
J -> I-> G -> D

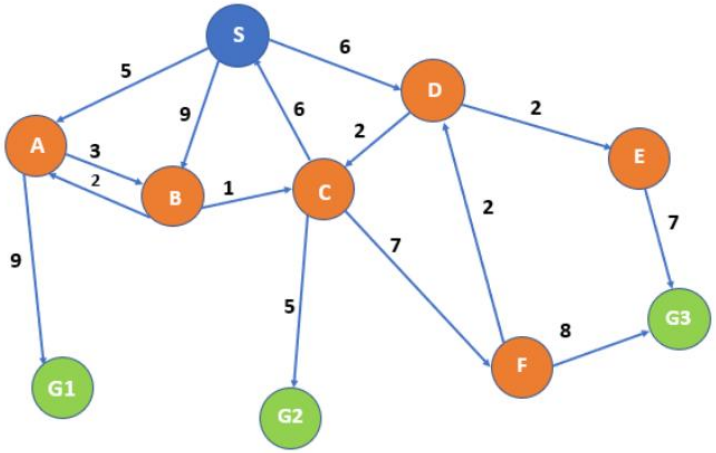
6. Uniform-cost Search Algorithm

- UCS is different from BFS and DFS because here the costs come into play. In other words, traversing via different edges might not have the same cost. Uniform-cost search is a searching algorithm used for **traversing a weighted tree or graph.**
- This algorithm comes into play when a **different cost is available for each edge**. The primary goal of the uniform-cost search is to find a path to the goal node which has the **lowest cumulative cost**.
- Uniform-cost search expands nodes according to their **path costs** form the root node. It can be used to solve any graph/tree where the optimal cost is in demand.
- A uniform-cost search algorithm is implemented by the **priority queue.** It gives maximum priority to the lowest cumulative cost.
- Uniform cost search is equivalent to **BFS** algorithm if the path cost of all **edges is the same**.


S
1
4
A
B
3
2
5
C
D
G
5
5
3
E
F
G
5
G


S
1
4
A
B
3
2
5
C
D
G
5
3
E
F
G
5
G


A
B
C
D
E
F
G1
G2
G3
1
2
3
4
1
1
3
2
2
5
2
2
3
1
2
2


S
A
B
C
D
E
F
G1
G2
G3
5
6
9
6
2
3
2
1
2
9
7
5
7
8

Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.

Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

Informed Search Algorithms

- Informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach to goal node, etc. This knowledge help agents to explore less to the search space and find more efficiently the goal node.
- Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.
- **Heuristics function:** Heuristic is a function which is used in Informed Search, and it finds the most promising path. **It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal.**
- The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time.
- The value of the heuristic function is always **positive**.

- Heuristic function is given as:

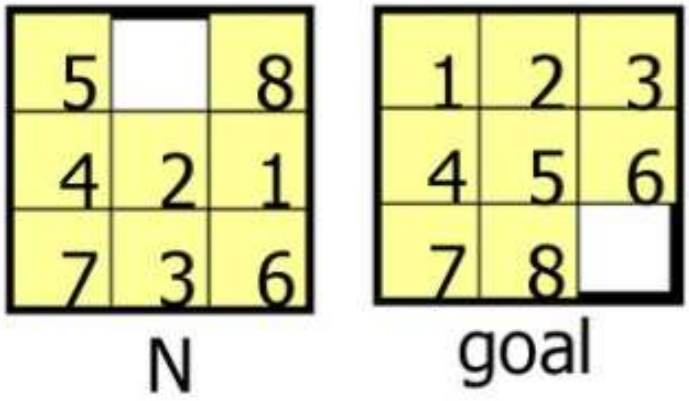
$$h(n) <= h^*(n)$$

- Here h(n) is heuristic cost, and h*(n) is the estimated cost. Hence heuristic cost should be less than or equal to the estimated cost.

Pure Heuristic Search

- Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based on their heuristic value h(n). It maintains two lists, **OPEN and CLOSED list**. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

- On each iteration, each node n with the **lowest heuristic** value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

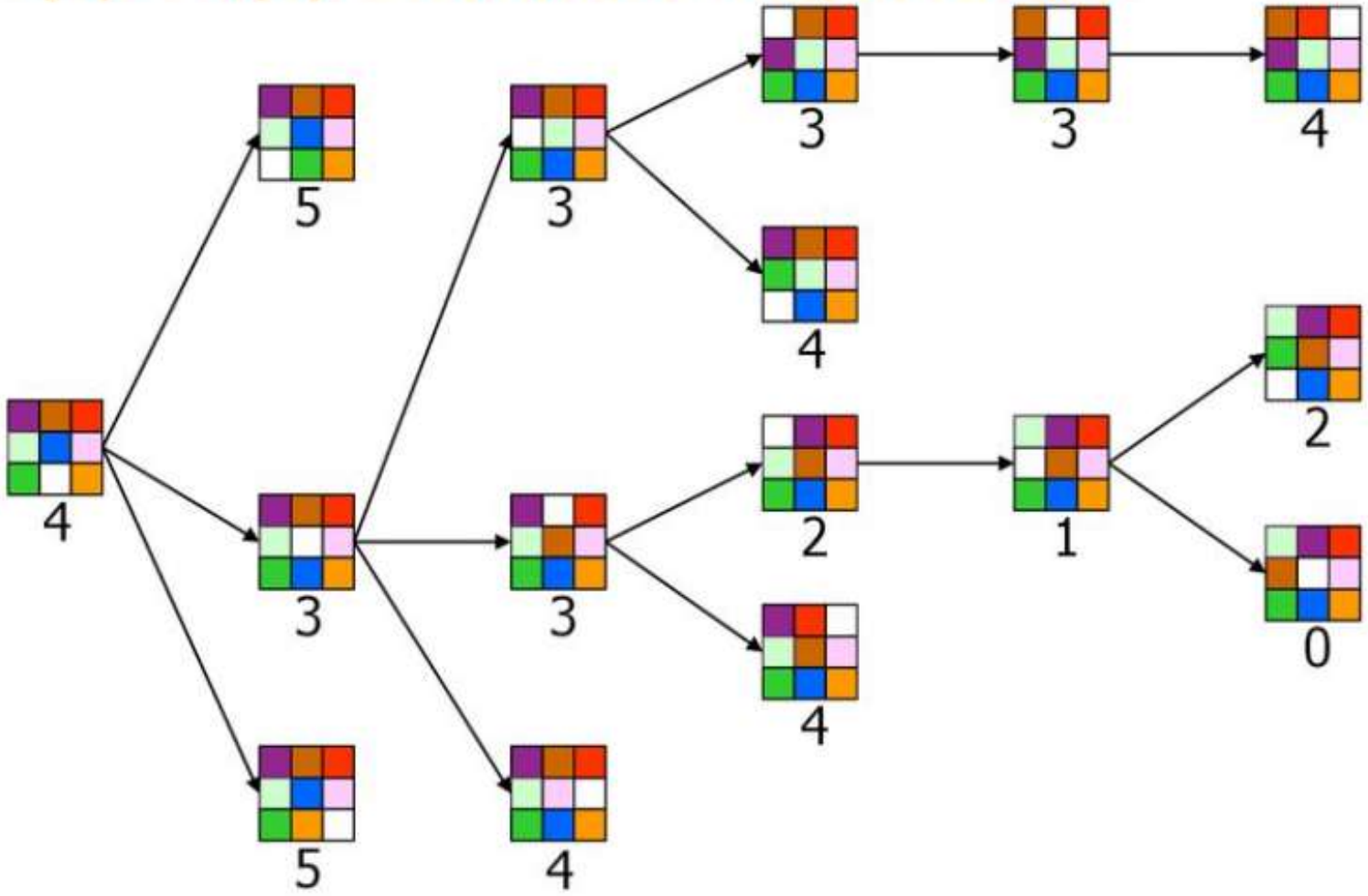


h(N) = number of misplaced tiles
= 6

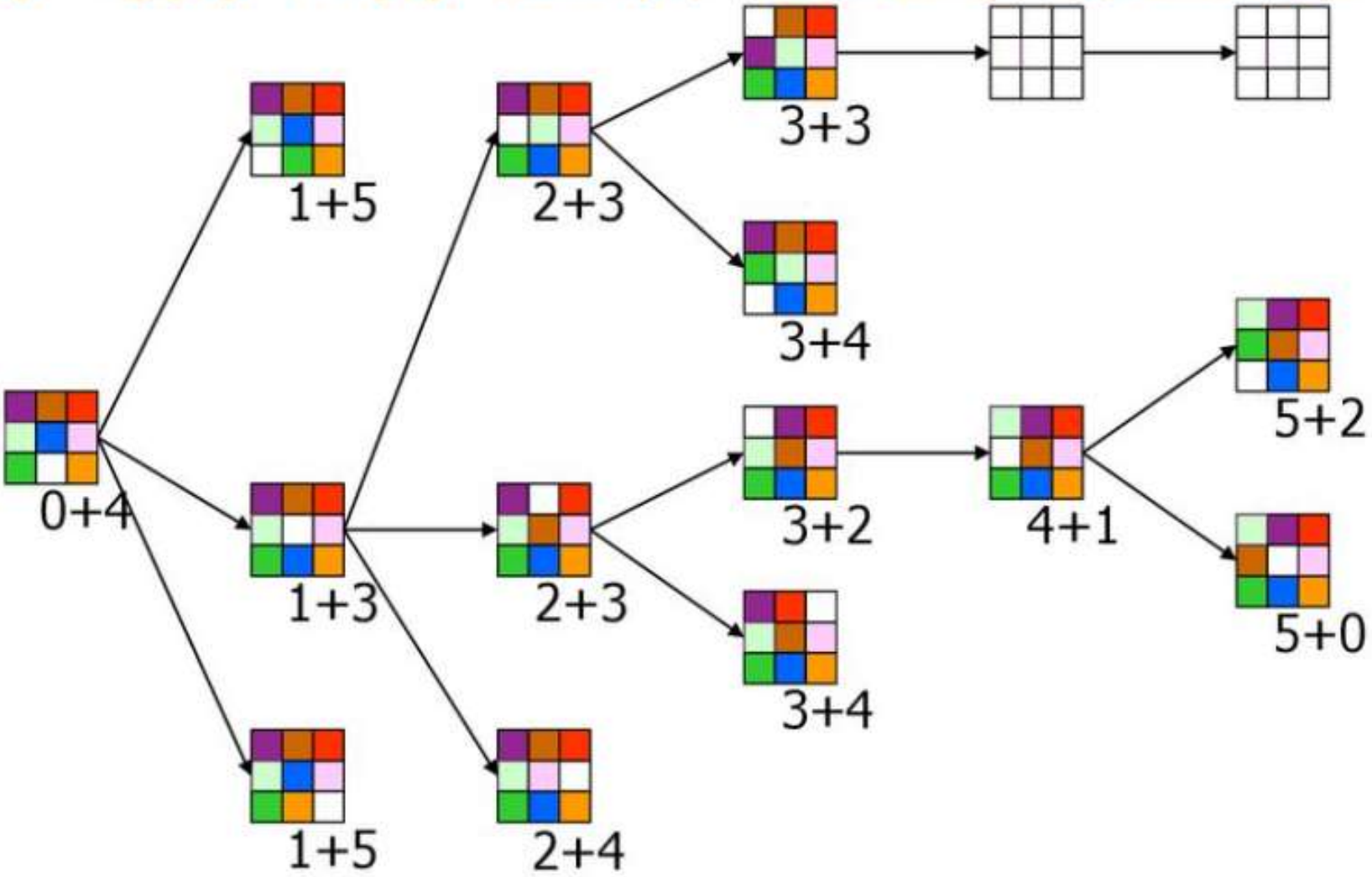
h(N) = sum of the distances of every tile to its goal position
= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1
= 13

8-PUZZLE

f(N) = h(N) = number of misplaced tiles



$f(N) = g(N) + h(N)$ with h(N) = number of misplaced tiles

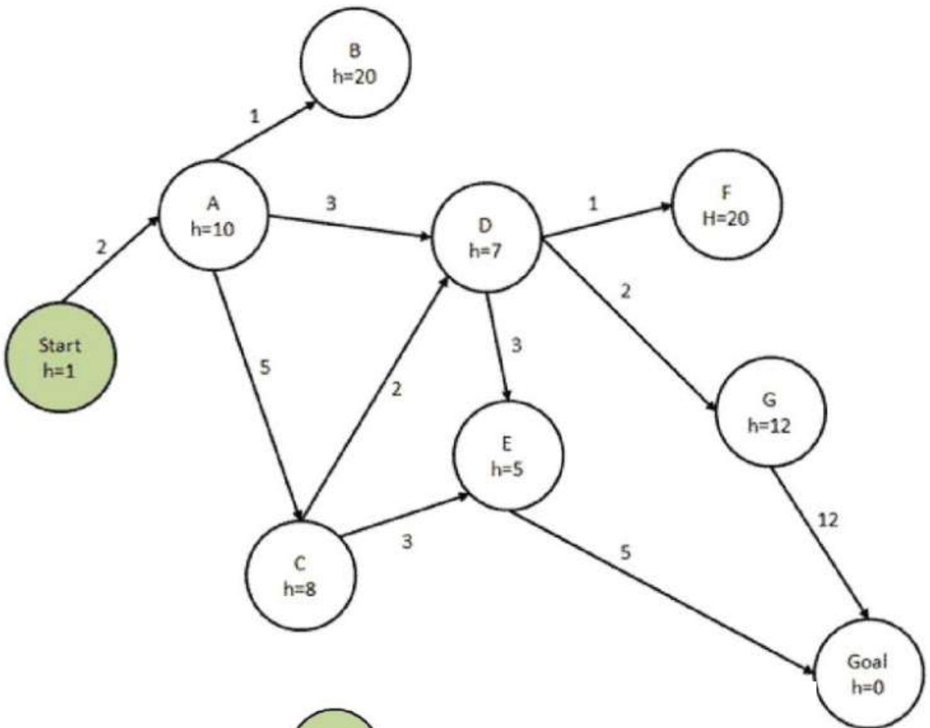


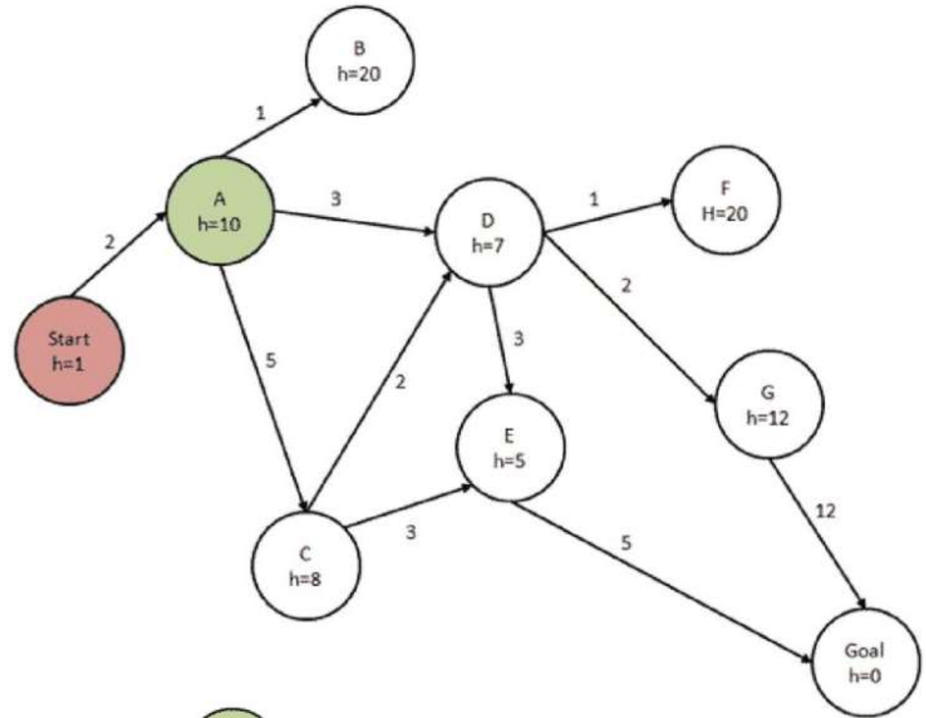
- In the informed search we will discuss two main algorithms

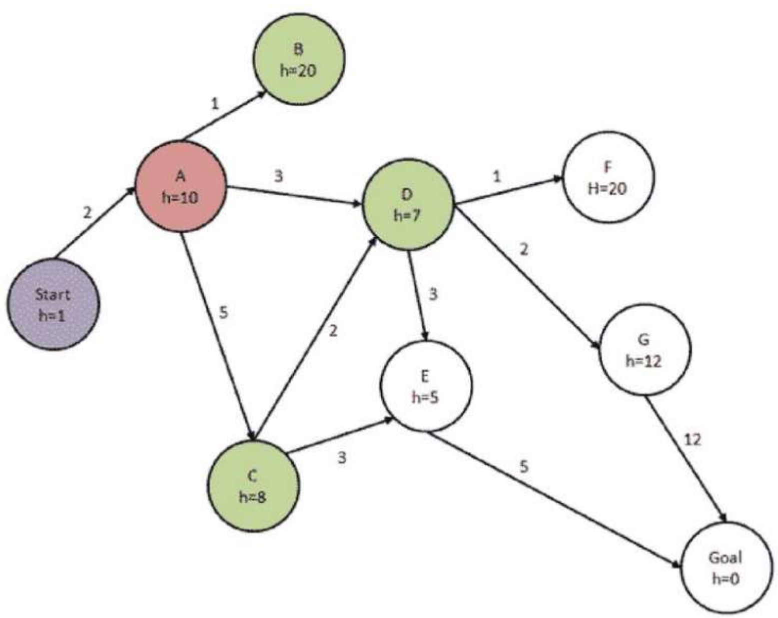
1) Best First Search Algorithm(Greedy search)
2) A* Search Algorithm

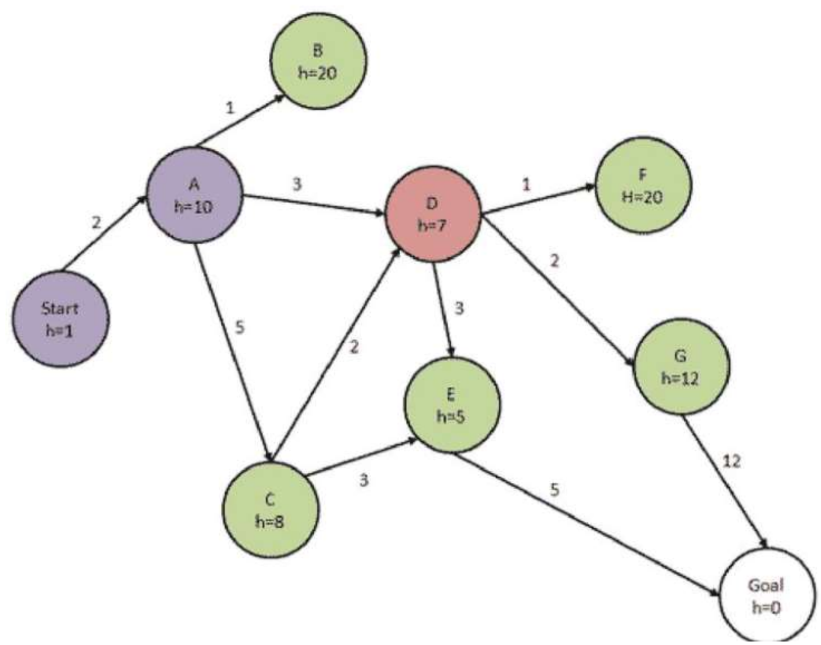
Best-first Search Algorithm (Greedy Search):

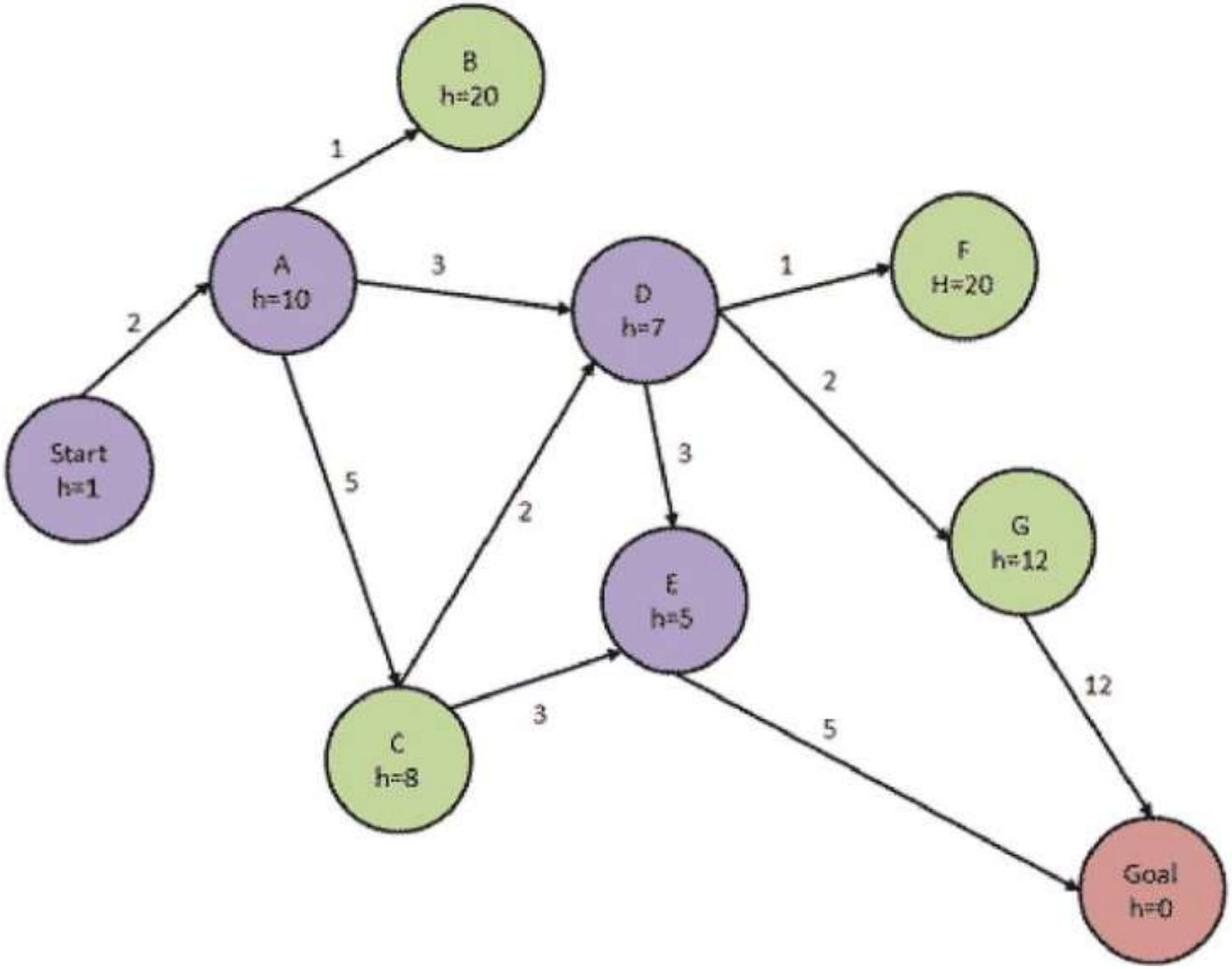
- Greedy best-first search algorithm always selects the path which appears best at that moment.
- It is the combination of **depth-first search and breadth-first** search algorithms. It uses the heuristic function and search.
- In the best first search algorithm, we expand the node **which is closest to the goal node**


Start h=1
A h=10
B h=20
C h=8
D h=7
E h=5
F H=20
G h=12
Goal h=0
2
1
3
5
2
1
2
3
3
5
12

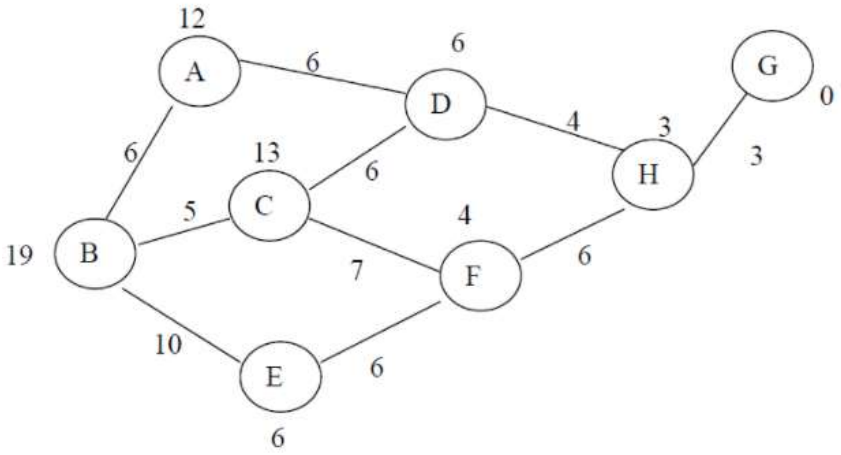

Start h=1
A h=10
B h=20
C h=8
D h=7
E h=5
F H=20
G h=12
Goal h=0
2
1
3
5
2
1
2
3
3
5
12

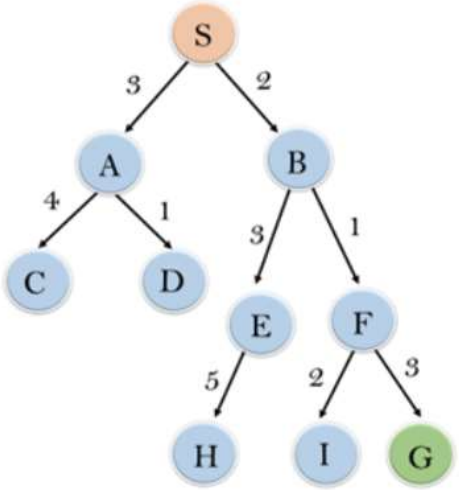

Start h=1
A h=10
B h=20
C h=8
D h=7
E h=5
F H=20
G h=12
Goal h=0
2
1
3
5
2
1
2
3
3
5
12


Start h=1
A h=10
B h=20
C h=8
D h=7
E h=5
F H=20
G h=12
Goal h=0
2
1
3
5
2
1
2
3
3
5
12



Start → A → D → E → Goal


12
A
6
6
D
G
0
4
3
3
6
13
6
H
5
C
4
19
B
7
F
6
10
E
6
6



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

Best First Search Algorithm:

Step 1: Place the starting node into the OPEN list.

Step 2: If the OPEN list is empty, Stop and return failure.

Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.

Step 4: Expand the node n, and generate the successors of node n.

Step 5: Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.

Step 6: For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.

Step 7: Return to Step 2.

Best First Search

```
Let fringe be a priority queue containing the initial state
Loop
    if fringe is empty return failure
    Node ← remove-first (fringe)
                if Node is a goal
                    then return the path from initial state to Node
      else generate all successors of Node, and
        put the newly generated nodes into fringe
        according to their f values
End Loop
```

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.

Space Complexity: The worst case space complexity of Greedy best first search is $O(b^m)$. Where, m is the maximum depth of the search space.

Complete: Greedy best-first search is also incomplete, even if the given state space is finite.

Optimal: Greedy best first search algorithm is not optimal.

Time Complexity

Space Complexity

Complete

Optimal

Advantages and Disadvantages of Best First Search

Advantages:

- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

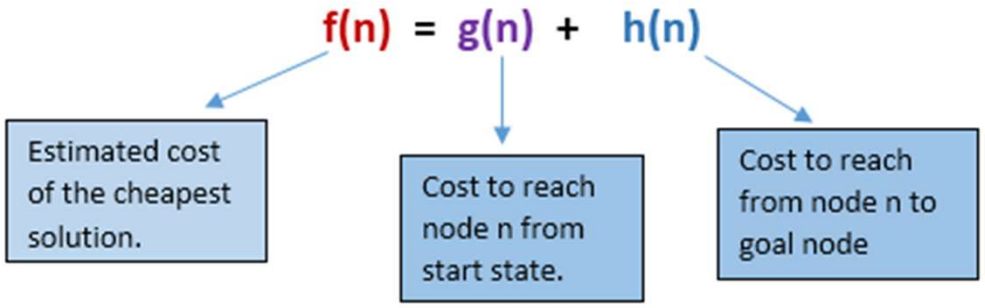
Disadvantages:

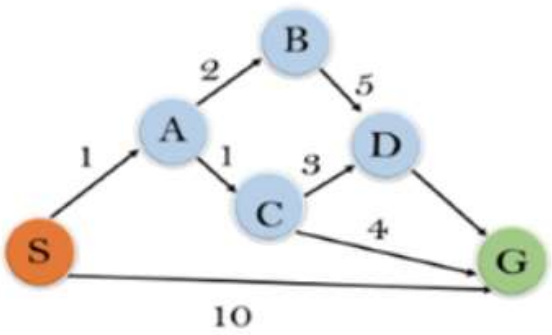
- It can behave as an unguided depth-first search in the worst case scenario.
- It can get stuck in a loop as DFS.
- This algorithm is not optimal.

A Search Algorithm*

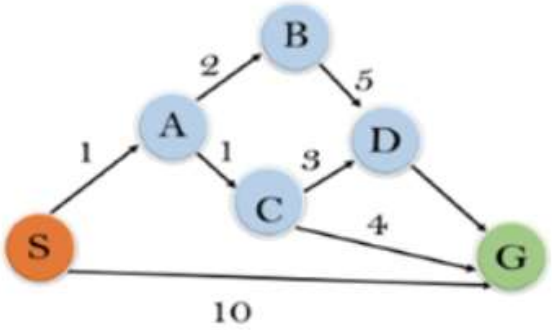
- A* search is the most commonly known form of best-first search.
- It uses **heuristic function h(n),** and **cost to reach the node n from the start state g(n)**.
- A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).
- It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree and provides **optimal result** faster.

- In A* search algorithm, we use search **heuristic** as well as the **cost to reach** the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.

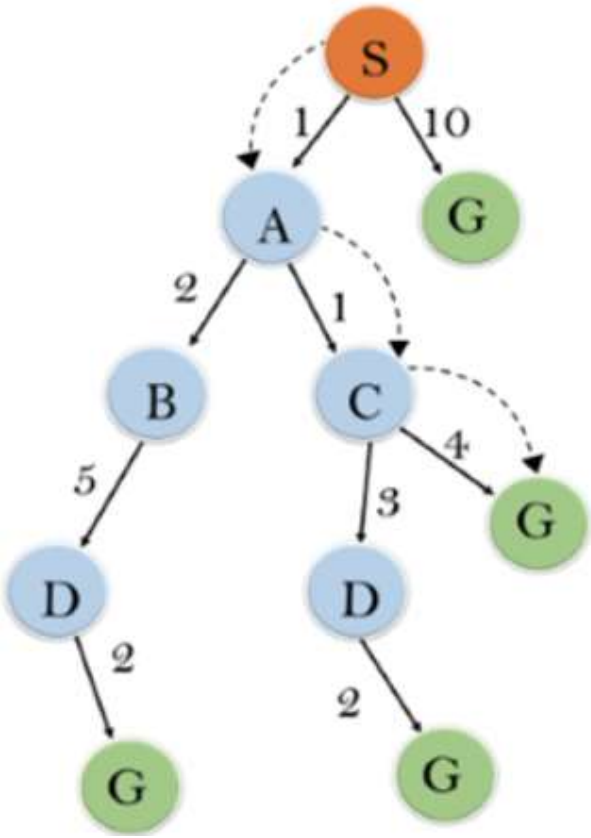




State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0



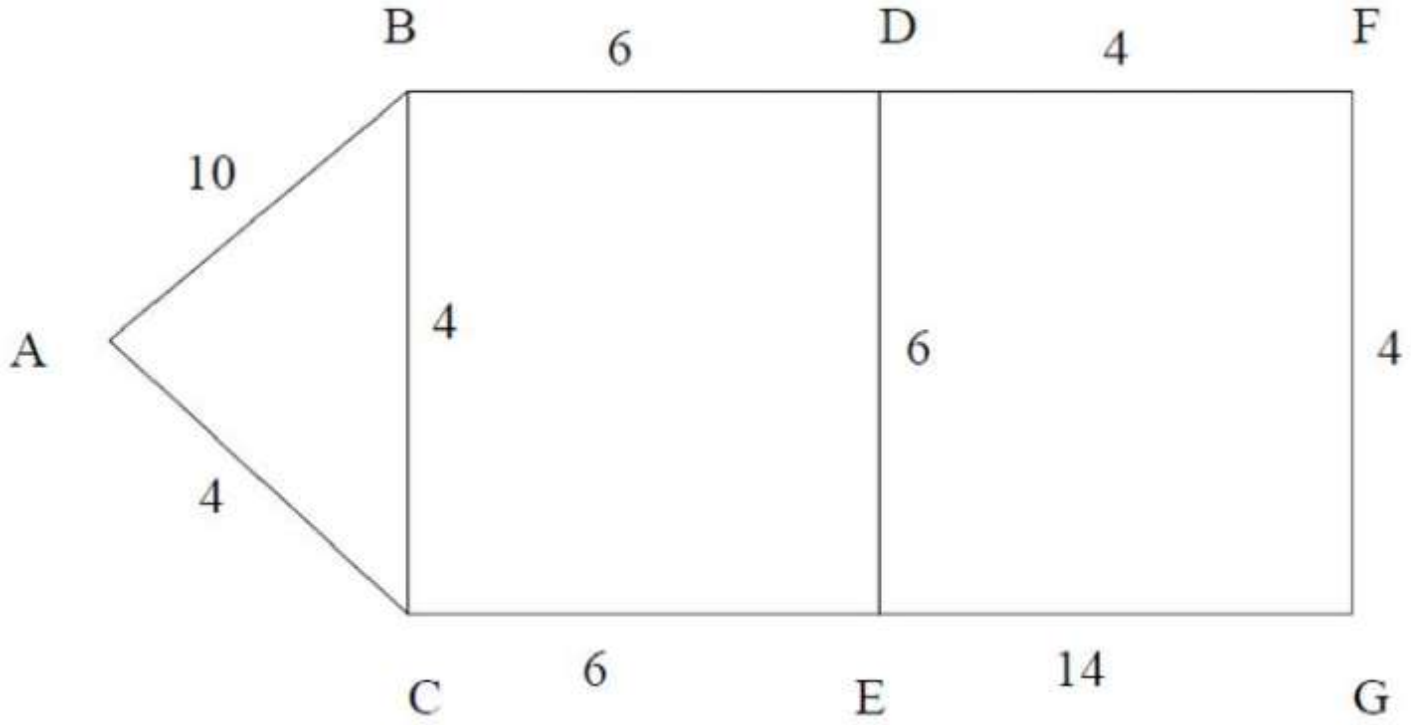
Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

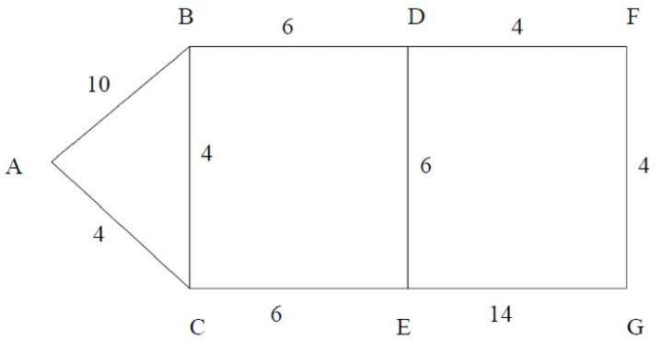
Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.



Assume h(n)=

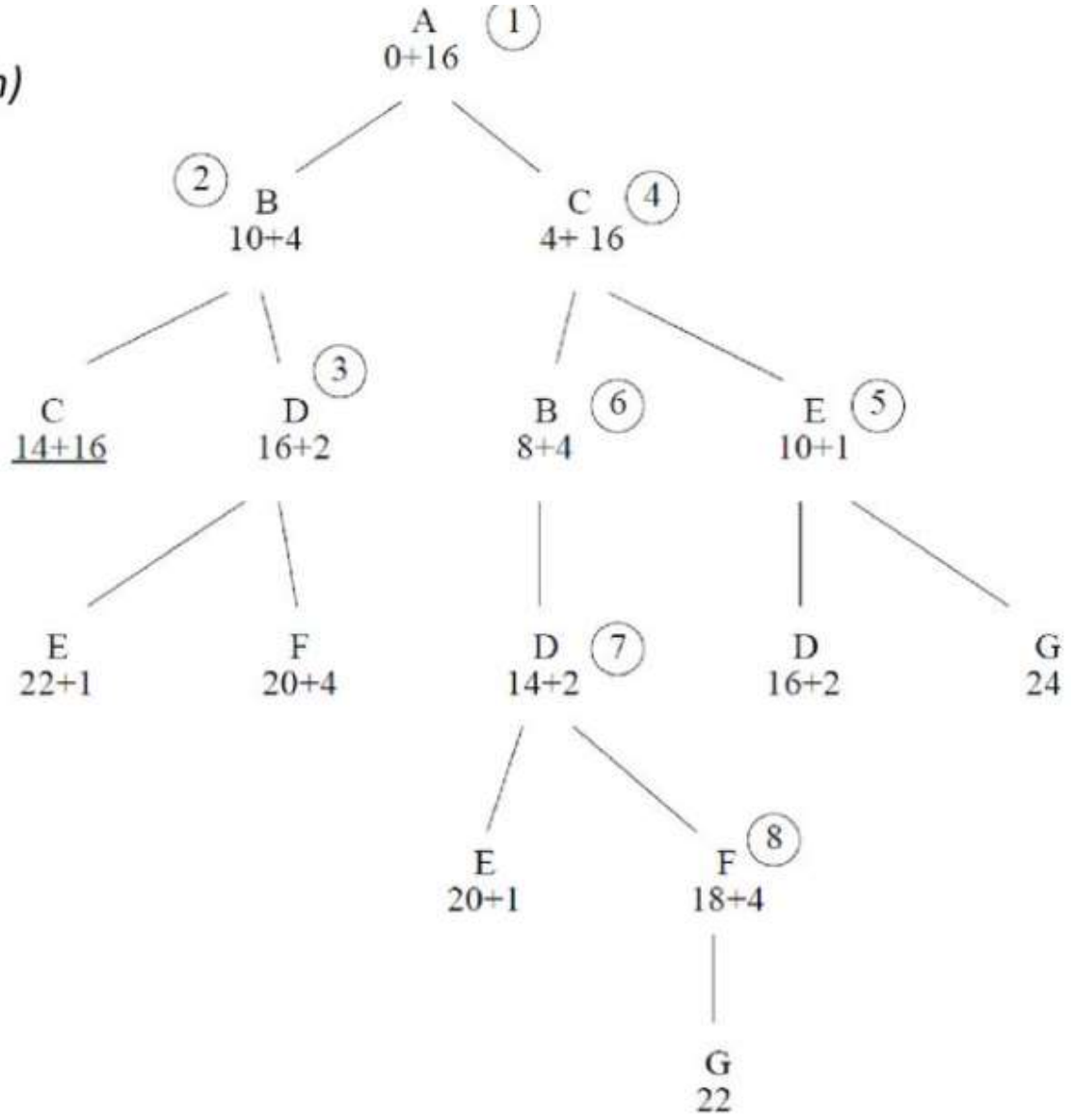
From A	16
From B	4
From C	16
From D	2
From E	1
From F	4

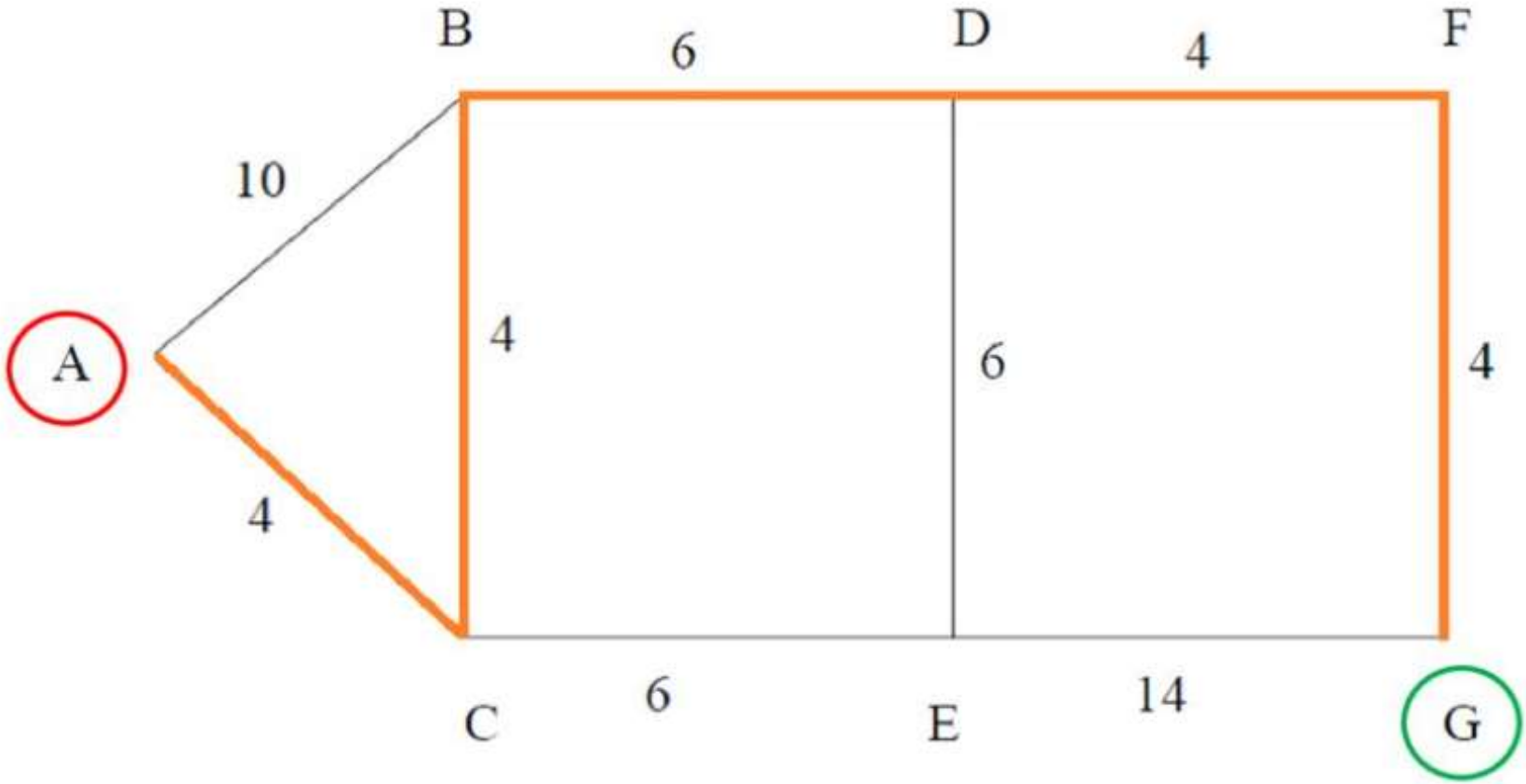
$$f(n) = g(n) + h(n)$$

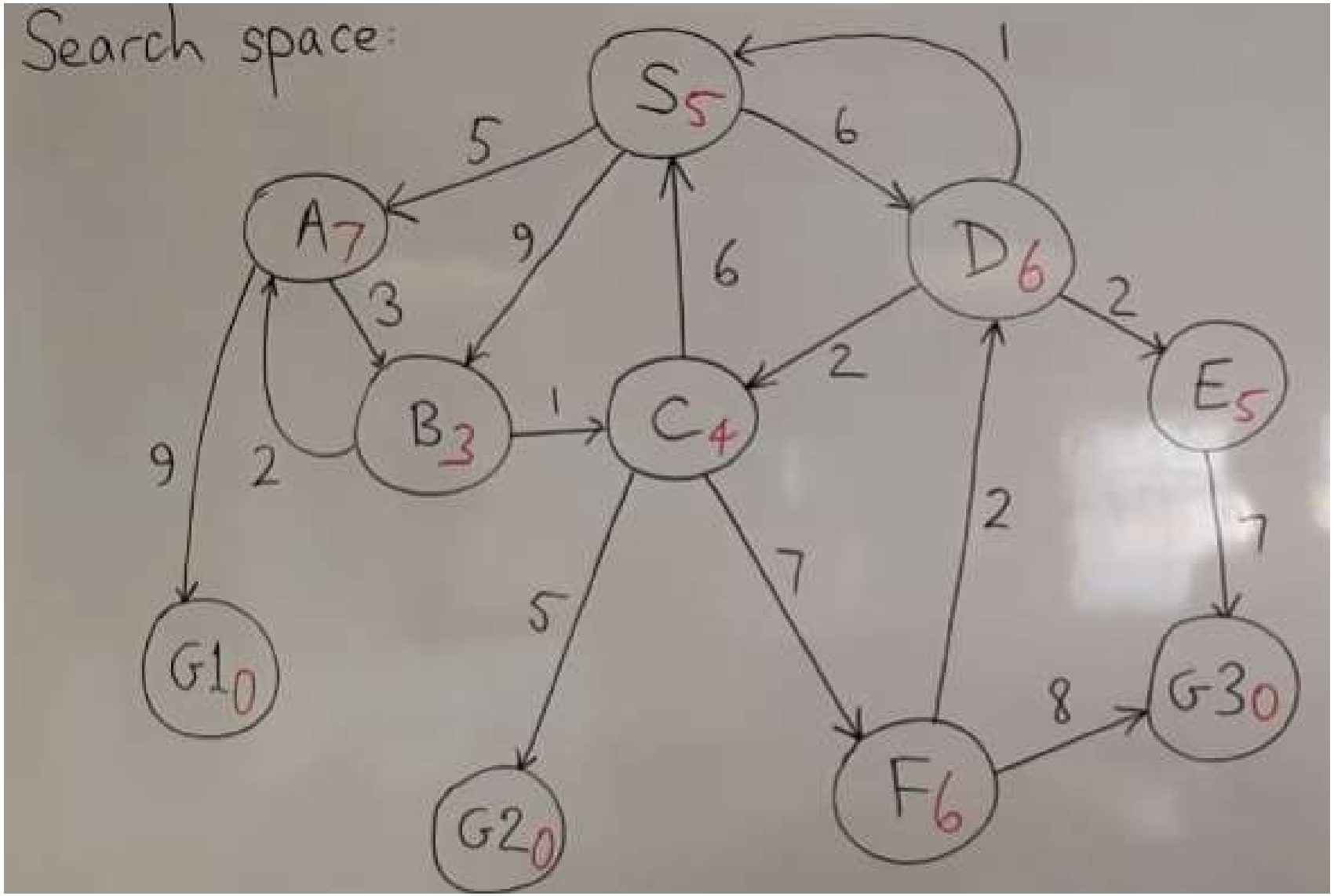


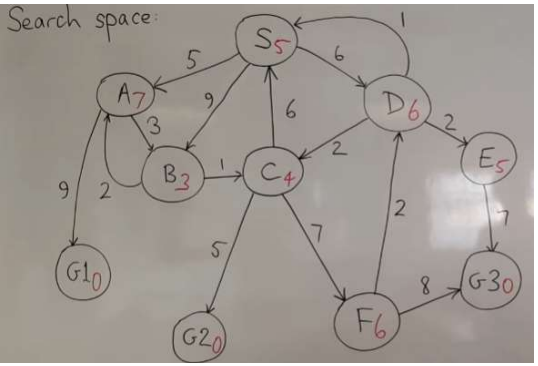
Assume h(n)=

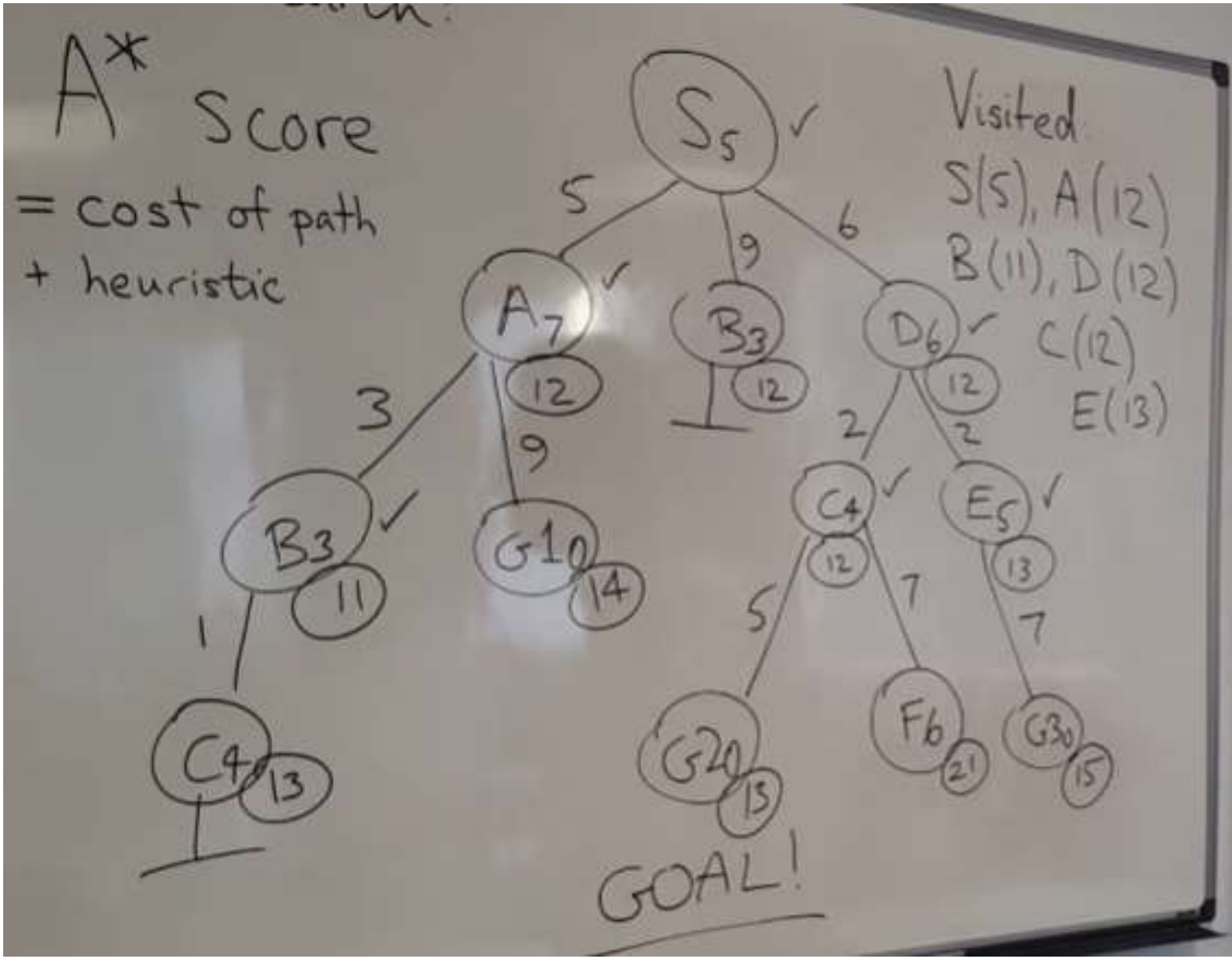
From A	16
From B	4
From C	16
From D	2
From E	1
From F	4

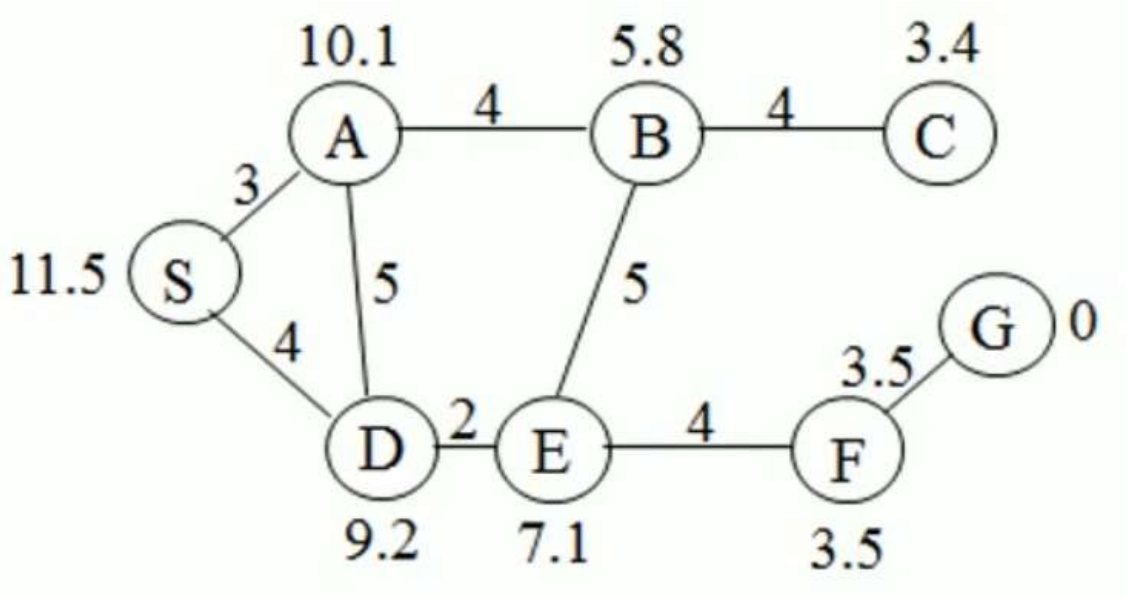


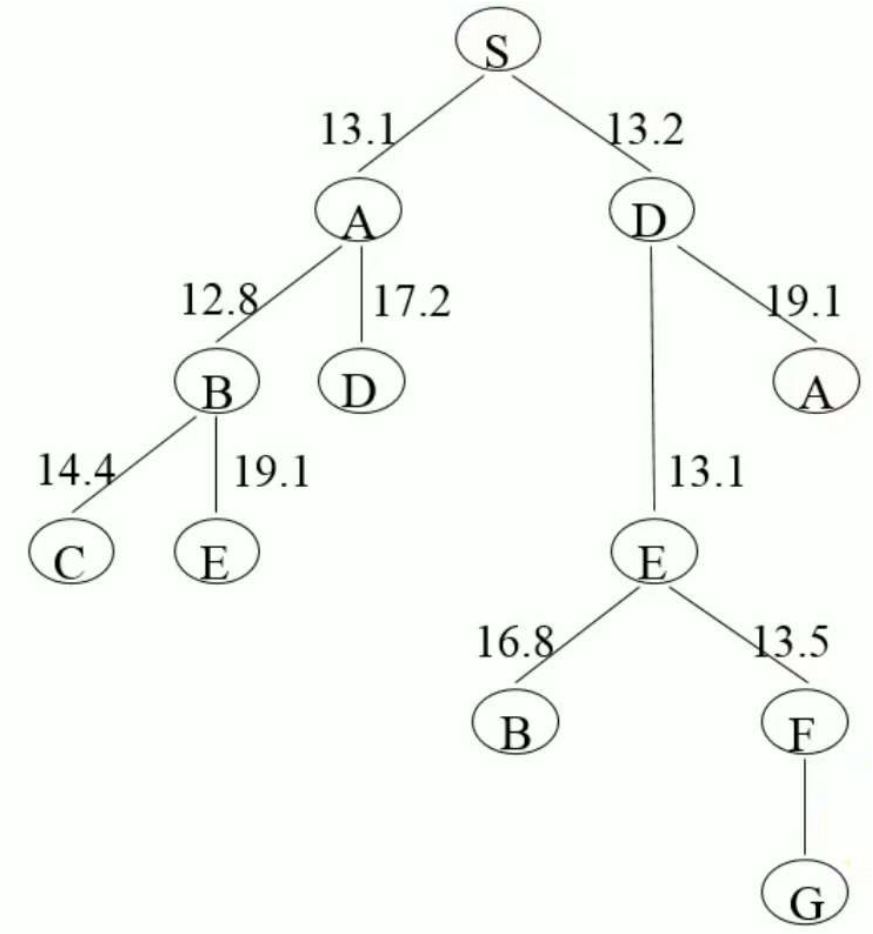

B
6
D
4
F
10
A
4
6
4
4
6
14
C
E
G

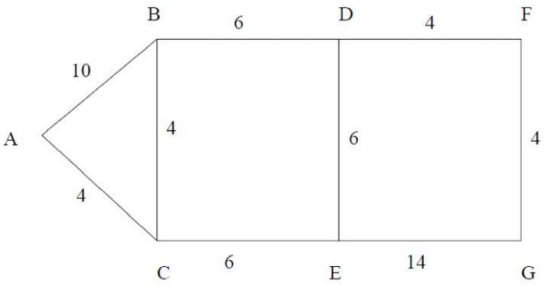

Search space:
S5
A7
D6
B3
C4
E5
G10
G20
F6
G30
1
5
6
9
6
3
2
1
9
2
2
7
5
7
8


Search space:
S5
A7
B3
C4
D6
E5
G10
G20
F6
G30


A* Score
= cost of path
+ heuristic
Visited
S(5), A(12)
B(11), D(12)
C(12)
E(13)
S5
A7
B3
D6
C4
E5
G10
G20
F6
G30
GOAL!


10.1
5.8
3.4
A
4
B
4
C
3
11.5
S
5
5
G
0
4
3.5
D
2
E
4
F
9.2
7.1
3.5


S
13.1
13.2
A
D
12.8
17.2
19.1
B
D
A
14.4
19.1
13.1
C
E
E
16.8
13.5
B
F
G



Algorithm A*

OPEN = nodes on frontier. CLOSED = expanded nodes.
OPEN = {<s, nil>}
while OPEN is not empty
    remove from OPEN the node <*n*,*p*> with minimum *f(n)*
    place <*n*,*p*> on CLOSED
    if *n* is a goal node,
        return success (path *p*)
    for each edge connecting *n* & *m* with cost *c*
        if <*m*, *q*> is on CLOSED and {*p*|*e*} is cheaper than *q*
            then remove *n* from CLOSED,
                put <*m*,{*p*|*e*}> on OPEN
        else if <*m*,*q*> is on OPEN and {*p*|*e*} is cheaper than *q*
            then replace *q* with {*p*|*e*}
        else if *m* is not on OPEN
            then put <*m*,{*p*|*e*}> on OPEN
return failure

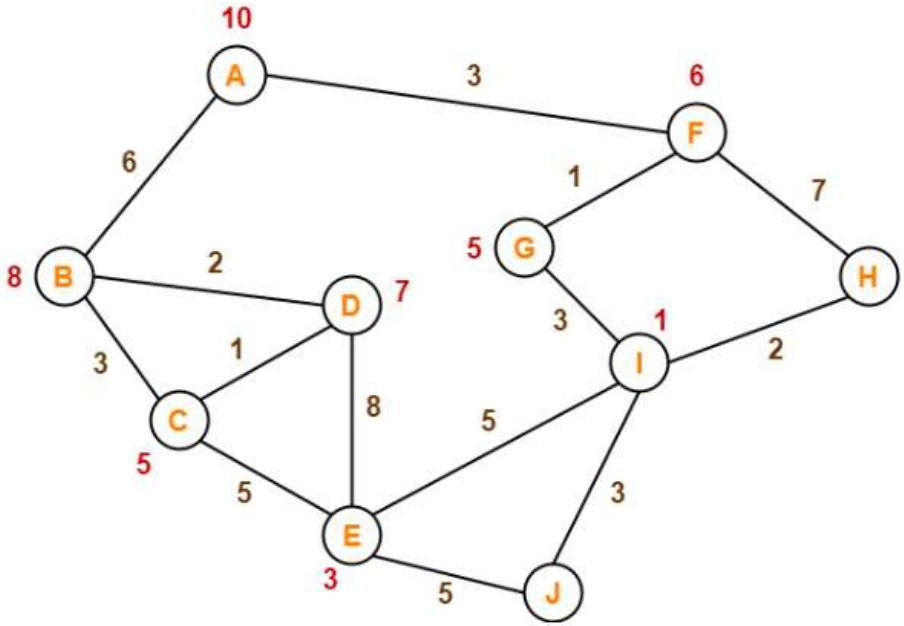
Advantages:

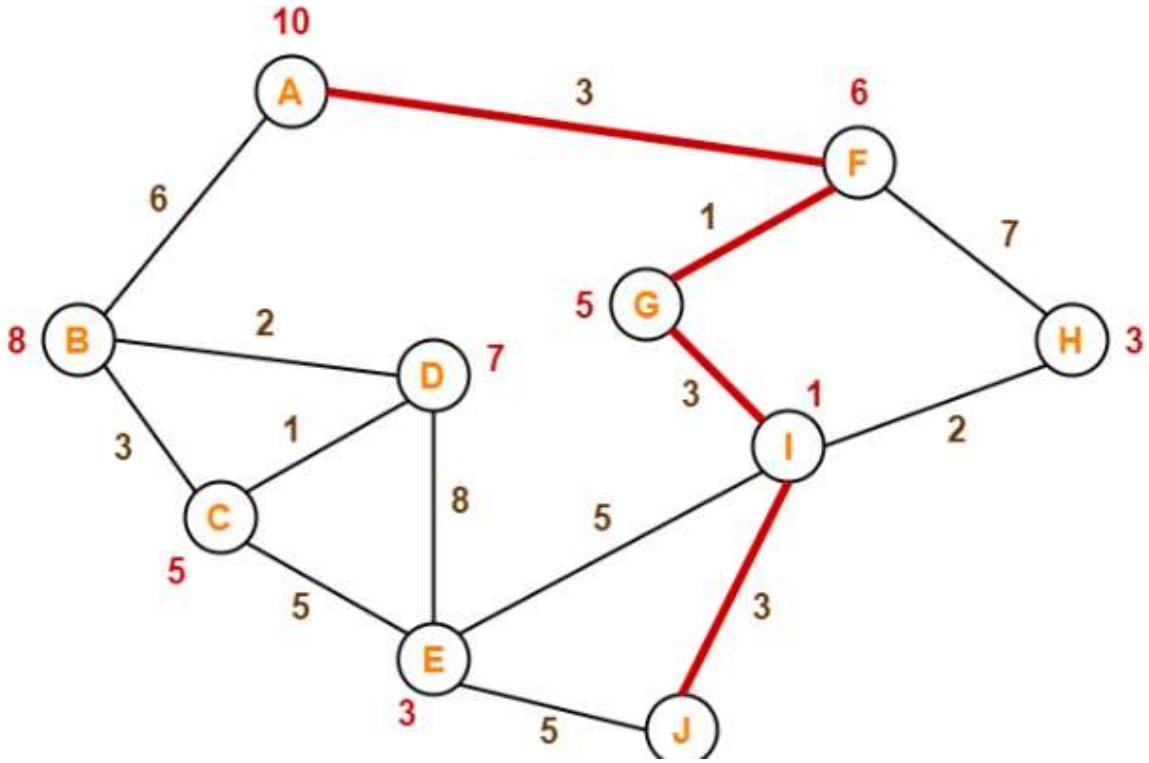
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Disadvantages:

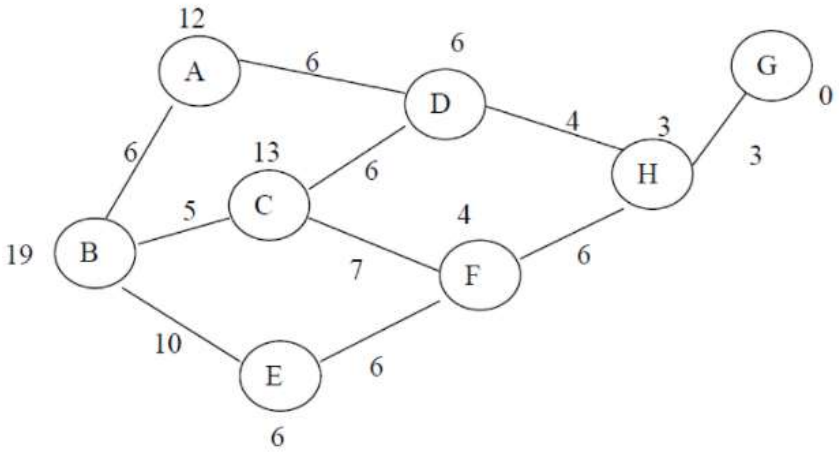
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

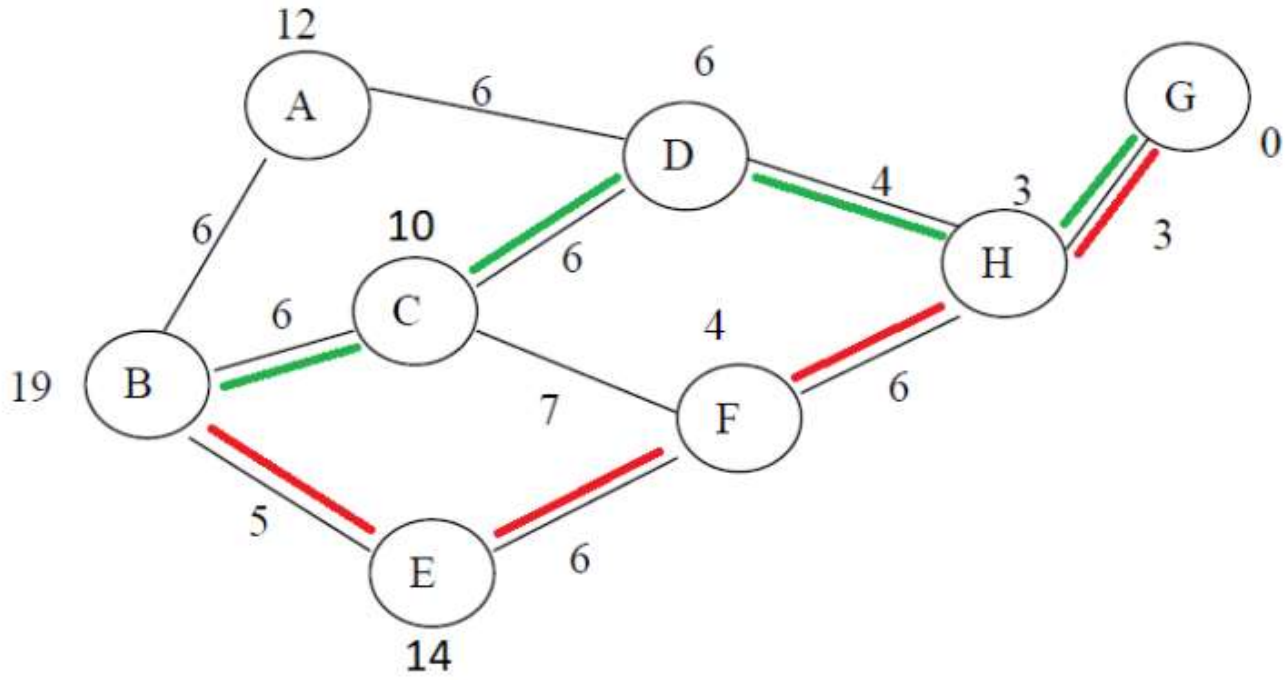
Find the most cost-effective path to reach from start state A to final state J using A* Algorithm.





Find the path by using Greedy Algorithm and Heuristic A* algorithm.





Path highlighted with red shows the path taken by Greedy Algorithm and the path highlighted with green shows the path taken by Heuristic A* algorithm.

8-Puzzle using A* Algorithm.

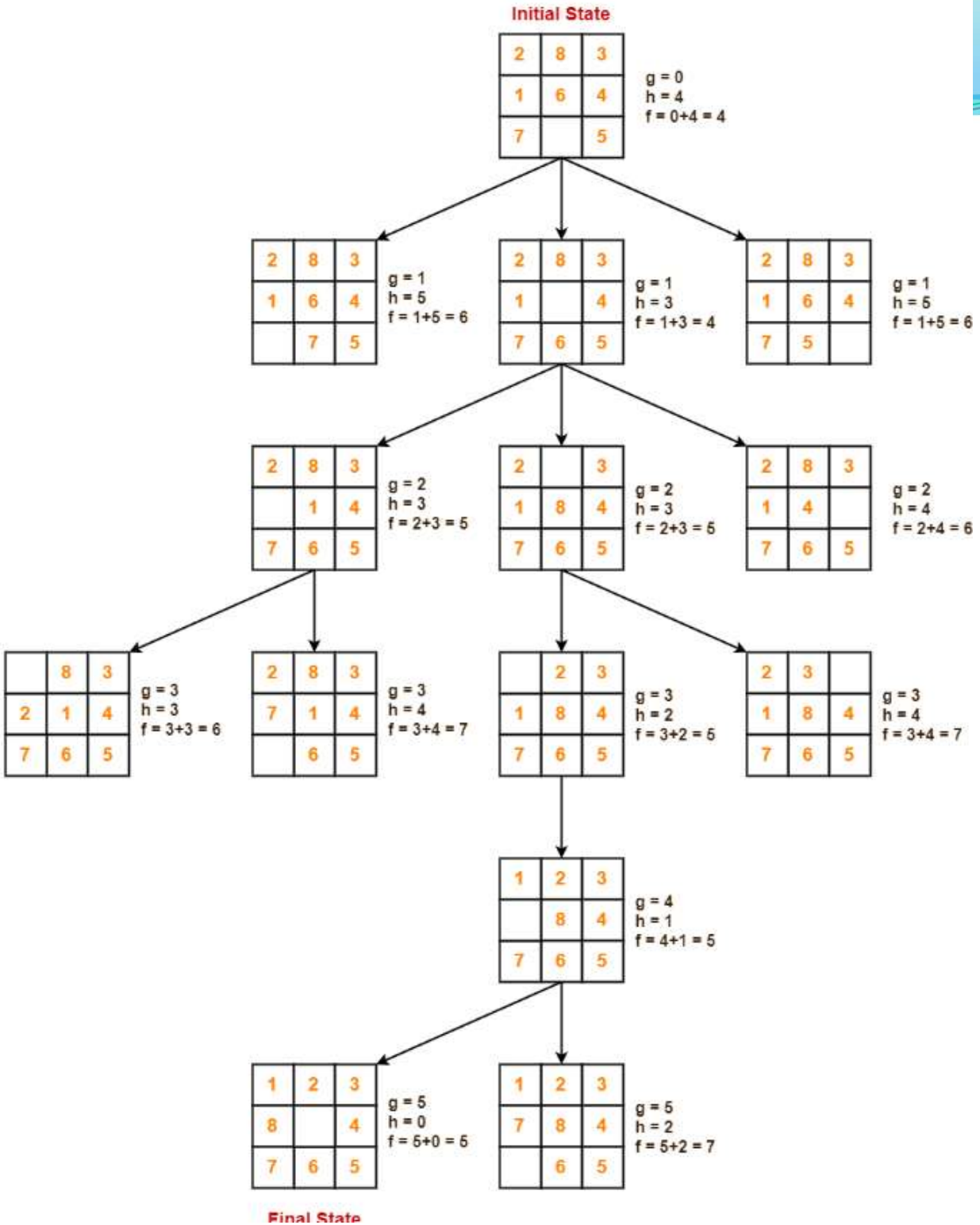
Find the most cost-effective path to reach the final state from initial state using A* Algorithm.

Initial State

Final State

$$F(n) = h(n) + g(n)$$

Consider g(n) = Depth of node
h(n) = Number of misplaced tiles.


Initial State
2 8 3
1 6 4
7 5
g = 0
h = 4
f = 0+4 = 4
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 1+5 = 6
2 8 3
1 4
7 6 5
g = 1
h = 3
f = 1+3 = 4
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 1+5 = 6
2 8 3
1 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
2 3
1 8 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
2 8 3
1 4
7 6 5
g = 2
h = 4
f = 2+4 = 6
8 3
2 1 4
7 6 5
g = 3
h = 3
f = 3+3 = 6
2 8 3
7 1 4
6 5
g = 3
h = 4
f = 3+4 = 7
2 3
1 8 4
7 6 5
g = 3
h = 2
f = 3+2 = 5
2 3
1 8 4
7 6 5
g = 3
h = 4
f = 3+4 = 7
1 2 3
8 4
7 6 5
g = 4
h = 1
f = 4+1 = 5
1 2 3
8 4
7 6 5
g = 5
h = 0
f = 5+0 = 5
1 2 3
7 8 4
6 5
g = 5
h = 2
f = 5+2 = 7
Final State

Hill Climbing

- Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem.
- It terminates when it reaches a **peak value** where no neighbor has a higher value.
- It is also called **greedy local search** as it only looks to its good immediate neighbor state and not beyond that.
- Hill Climbing is mostly used when a good heuristic is available.
- In this algorithm, we **don't need to maintain and handle the search tree or graph** as it only keeps a single current state.

Features of Hill Climbing

Greedy approach: Hill-climbing algorithm search moves in the direction which optimizes the cost.

No backtracking: It does not backtrack the search space, as it does not remember the previous states.

Feedback Mechanism: The feedback from the previous computation helps in deciding the next course of action i.e. whether to move up or down the slope

Types of Hill Climbing Algorithm:

1) Simple hill Climbing:
2) Steepest-Ascent hill-climbing:
3) Stochastic hill Climbing:

1. Simple Hill Climbing:

- Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state**. It only checks it's one successor state, and if it finds better than the current state, then move else be in the same state.

This algorithm has the following features:

- Less time consuming
- Less optimal solution and the solution is not guaranteed

Algorithm for Simple Hill Climbing:

Step 1: Evaluate the initial state, if it is goal state then return success and Stop.

Step 2: Loop Until a solution is found or there is no new operator left to apply.

Step 3: Select and apply an operator to the current state.

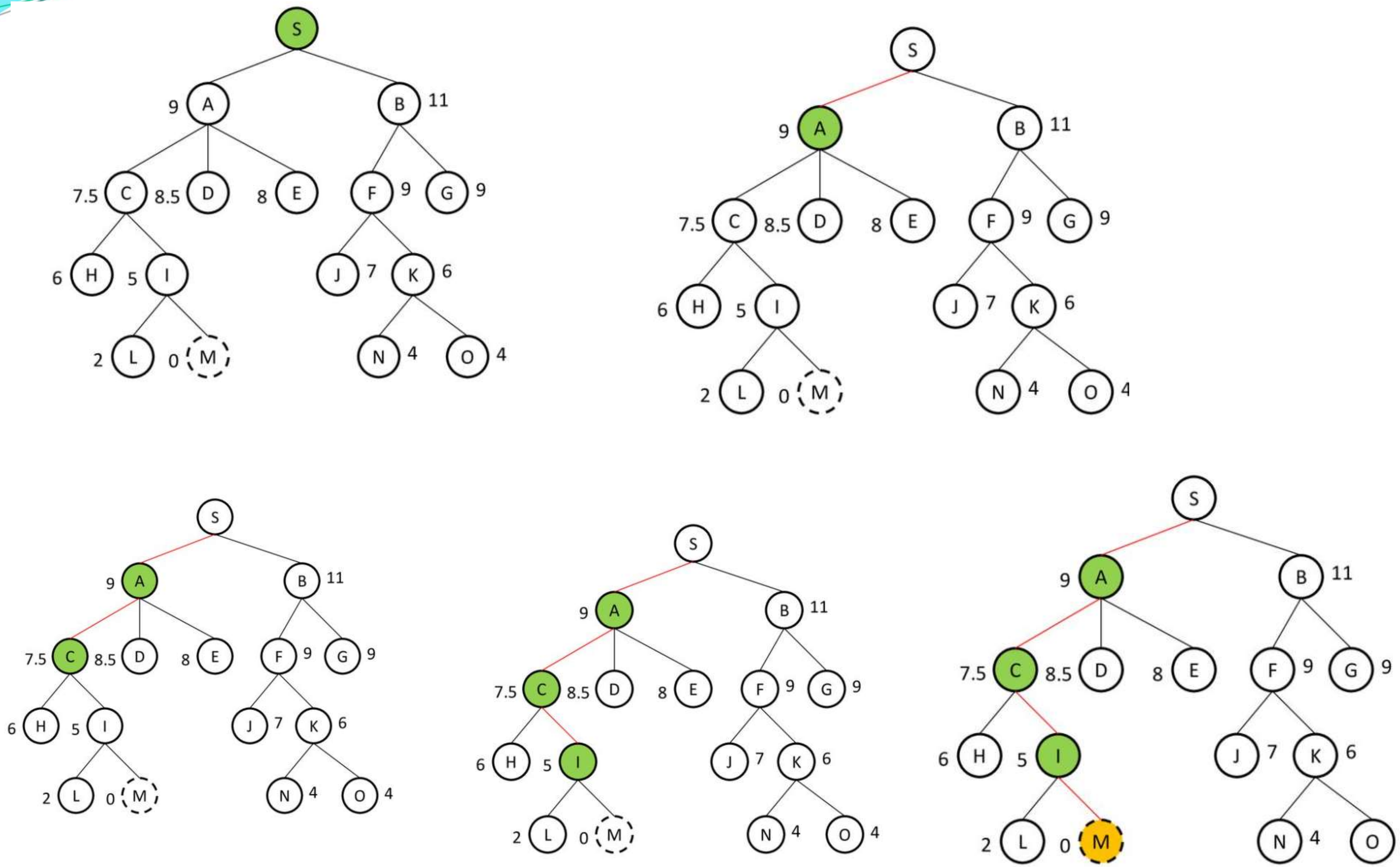
Step 4: Check new state:

If it is goal state, then return success and quit.

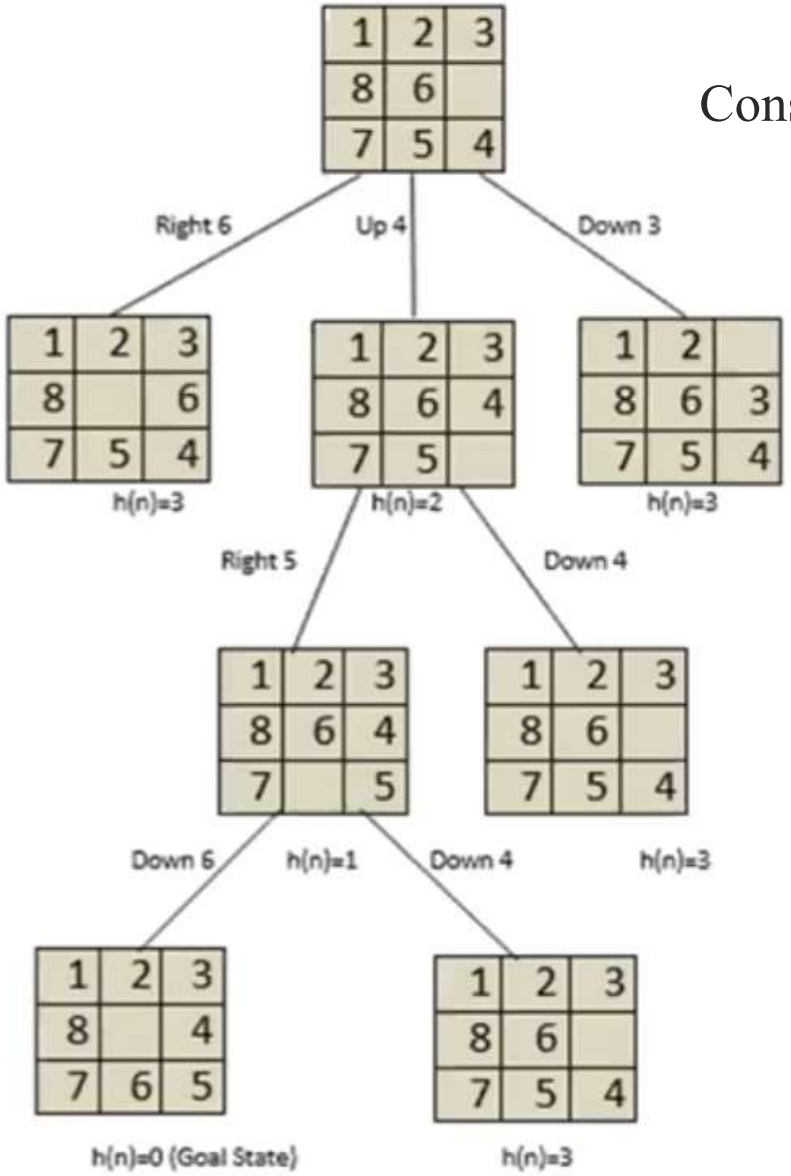
Else if it is better than the current state then assign new state as a current state.

Else if not better than the current state, then return to step2.

Step 5: Exit.


S
9 A
B 11
7.5 C
8.5 D
8 E
F 9
G 9
6 H
5 I
J 7
K 6
2 L
0 M
N 4
O 4

8 Puzzle Problem using Hill Climbing



Consider h(n) = Number of misplaced tiles.

2. Steepest-Ascent hill-climbing:

- Steepest-Ascent Hill-Climbing algorithm (gradient search) is a variant of Hill Climbing algorithm.

- In the case of hill climbing technique we picked any state as a successor which was closer to the goal than the current state whereas, in Steepest-Ascent Hill Climbing algorithm, we choose the best successor among all possible successors and then update the current state.

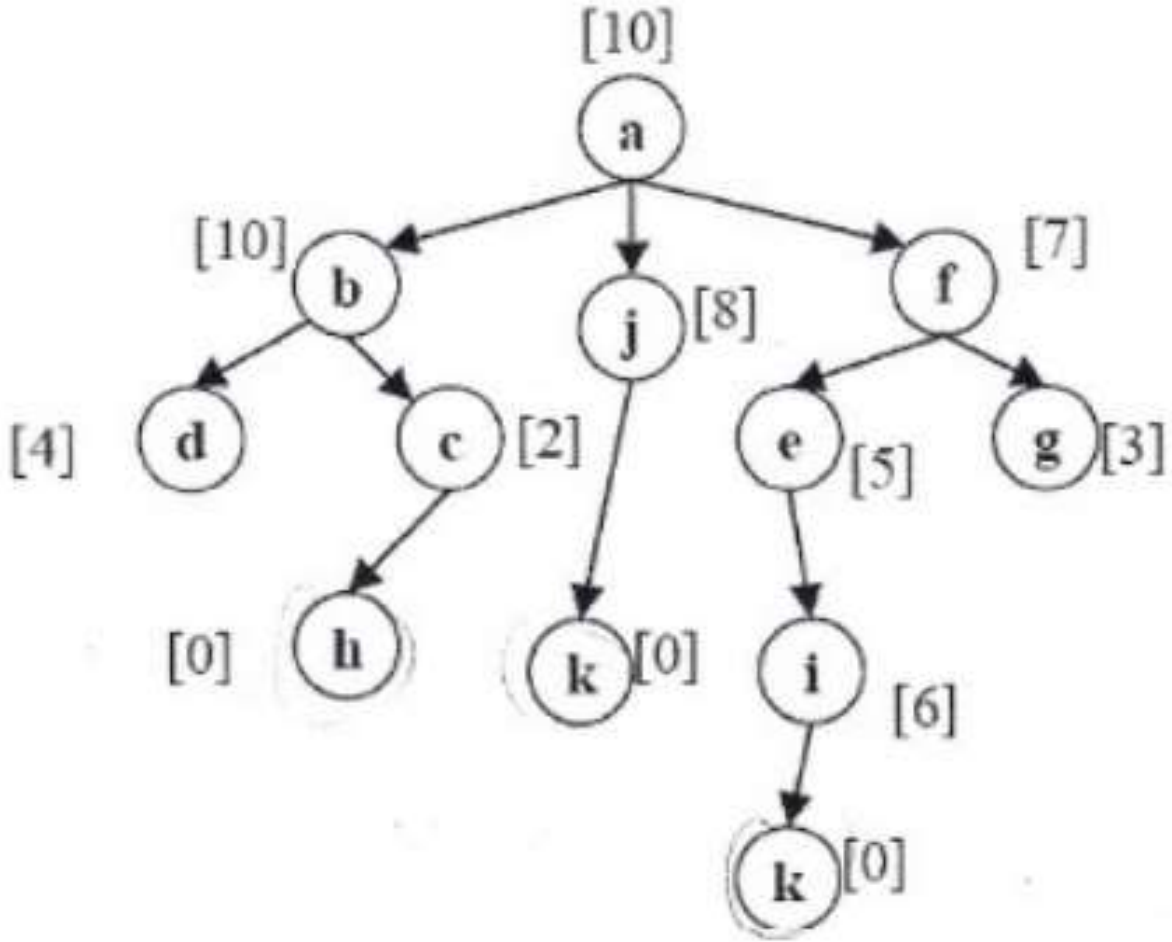
Algorithm for Steepest-Ascent hill climbing

- **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
- **Step 2:** Loop until a solution is found or the current state does not change.
    a. Let SUCC be a state such that any successor of the current state will be better than it.
    b. For each operator that applies to the current state:
        a. Apply the new operator and generate a new state.
        b. Evaluate the new state.
        c. If it is goal state, then return it and quit, else compare it to the SUCC.
        d. If it is better than SUCC, then set new state as SUCC.
        e. If the SUCC is better than the current state, then set current state to SUCC.
- **Step 5:** Exit.

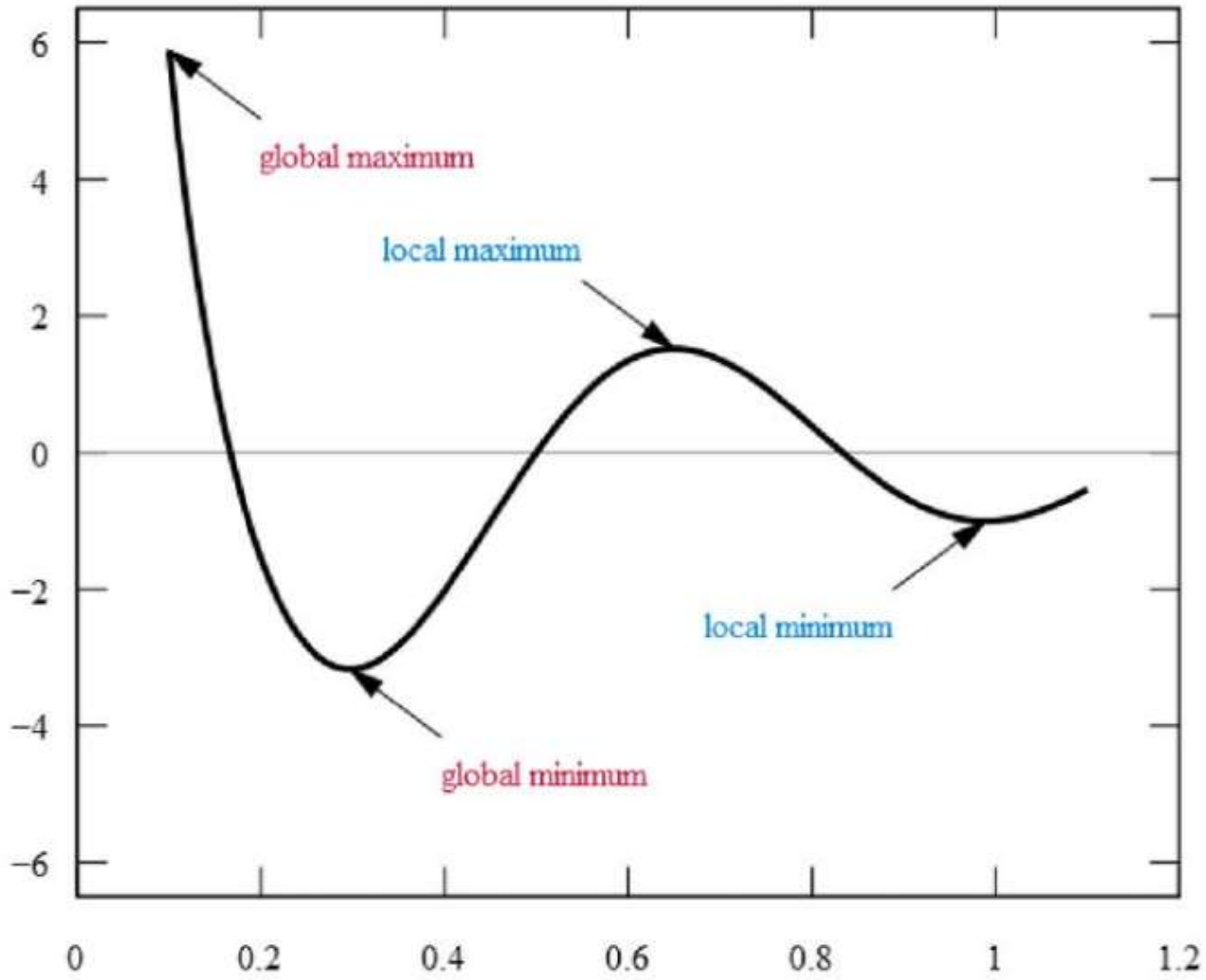
3. Stochastic hill climbing

- Stochastic hill climbing chooses at random from among the uphill moves.
- Stochastic hill climbing does not examine for all its neighbor before moving. Rather, this search algorithm **selects one neighbor node at random and decides whether** to choose it as a current state or examine another state.
- The probability of selection can vary with the steepness of the uphill move.
- Stochastic hill climbing usually converges more slowly than steepest ascent, but in some state landscapes, it finds better solutions.
- Stochastic hill climbing is NOT complete, but it may be less likely to get stuck.

Hill Climbing Algorithm Example



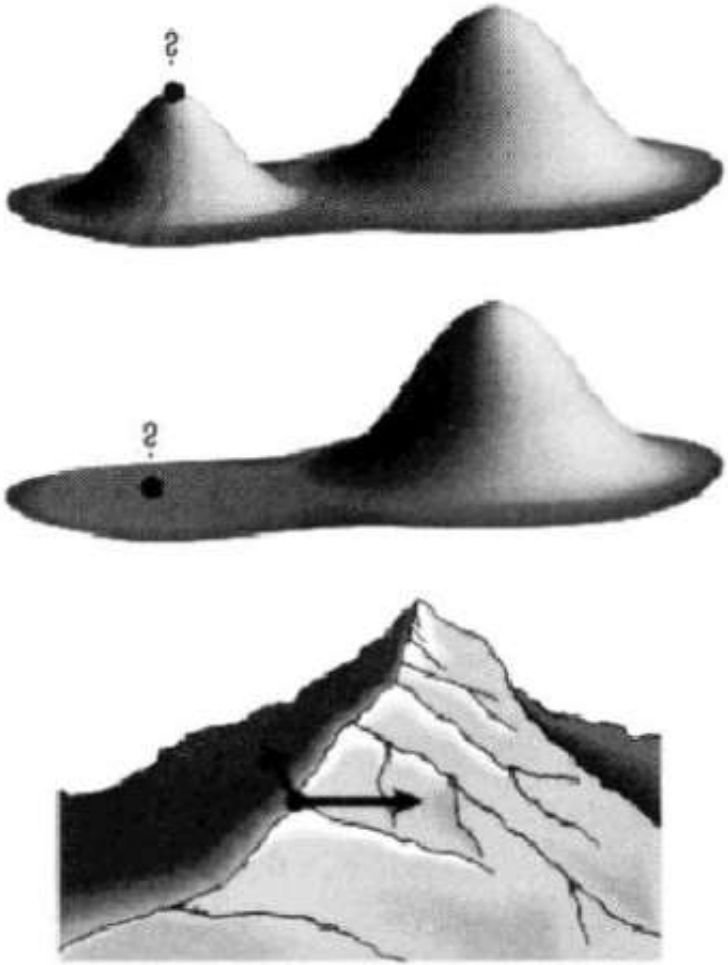
Local Maximum and Local Minimum

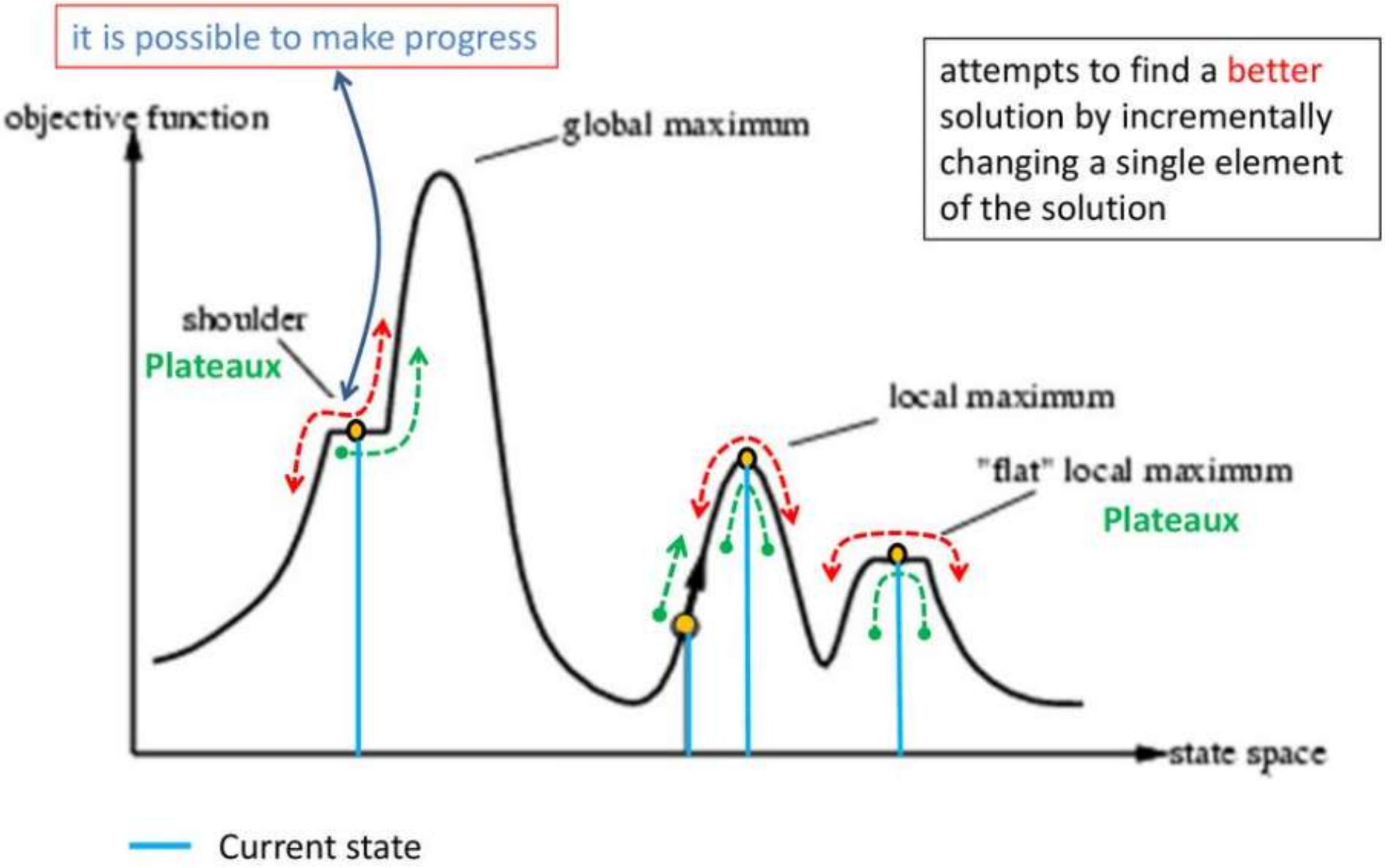


Local maxima: a local maximum is a peak that is higher than each of its neighboring states, but lower than the global maximum. Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn upwards towards the peak, but will then be stuck with nowhere else to go.

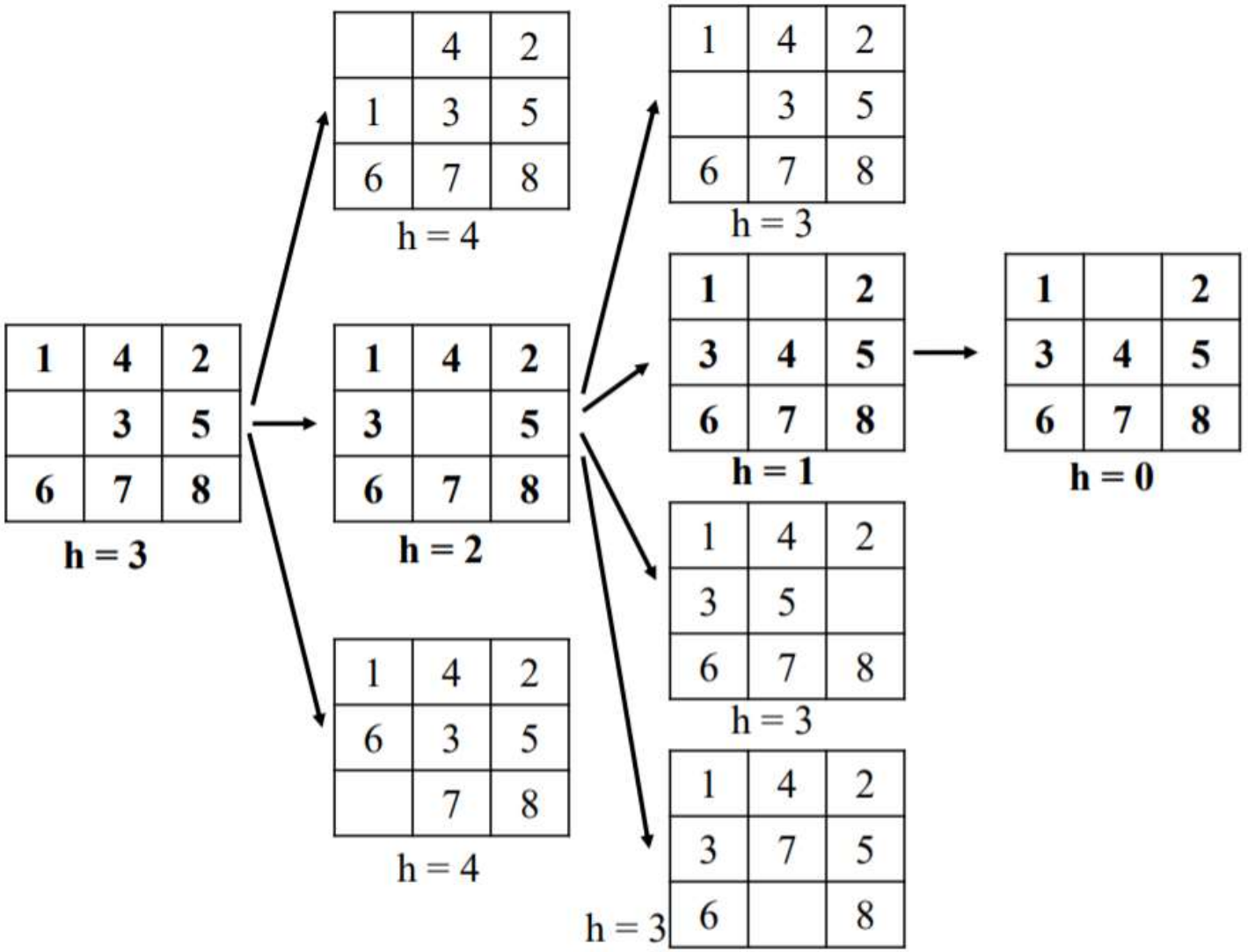
Plateaux: a plateau is an area of the state space landscape where the evaluation function is flat. It can be a flat local maximum, from which no uphill exit exists, or a **shoulder,** from which it is possible to make progress.

Ridges: Ridges result in a sequence of local maxima that is very difficult for greedy algorithms to navigate. (the search direction is not towards the top but towards the side)

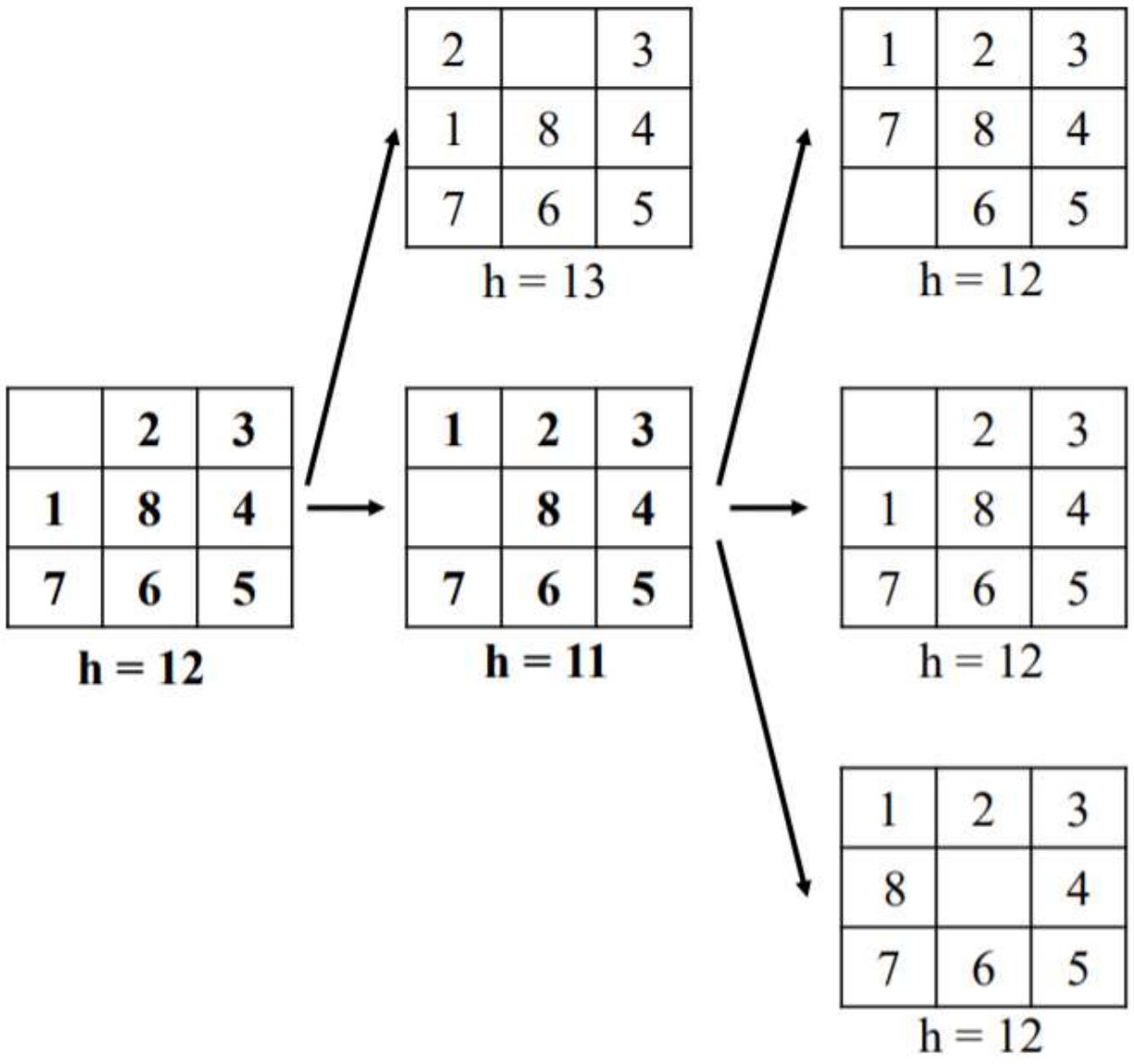



it is possible to make progress
attempts to find a better solution by incrementally changing a single element of the solution
objective function
global maximum
shoulder
Plateaux
local maximum
"flat" local maximum
Plateaux
state space
Current state

8-puzzle: a solution case



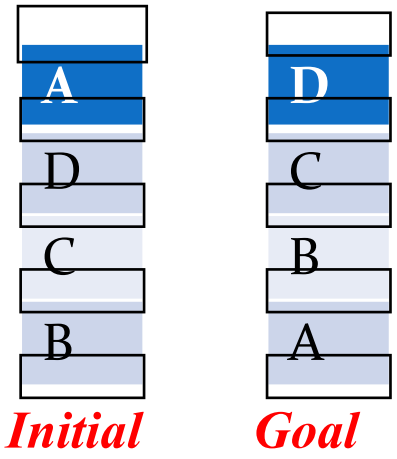
8-puzzle: stuck at local maximum

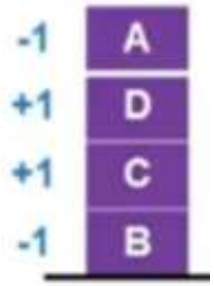


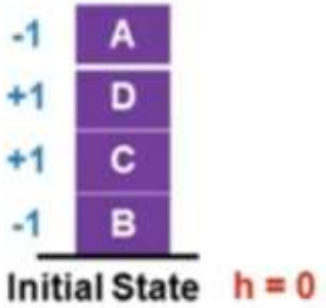
Blocks World Problem with Local Heuristic Function

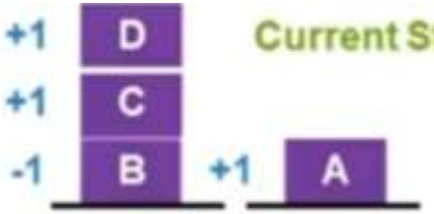
+1 for each block that is resting on the thing it is suppose to be resting on.

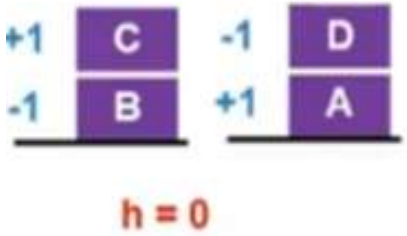
-1 for each block that is resting on wrong thing.

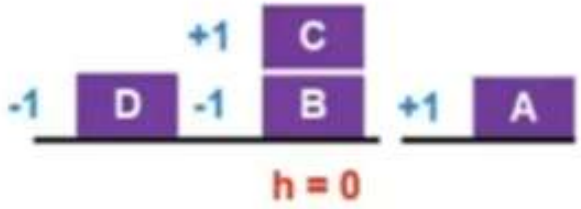


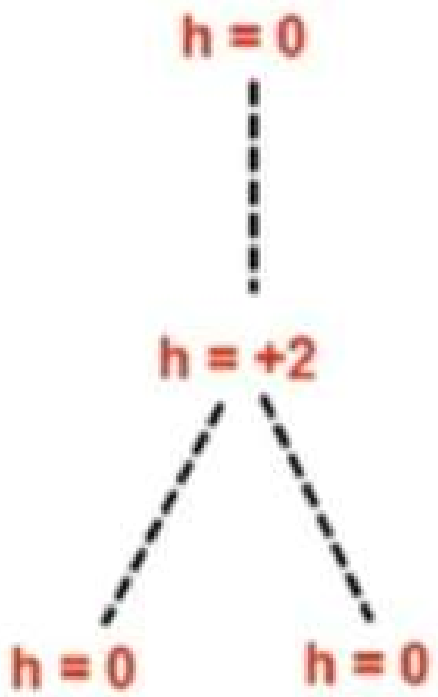


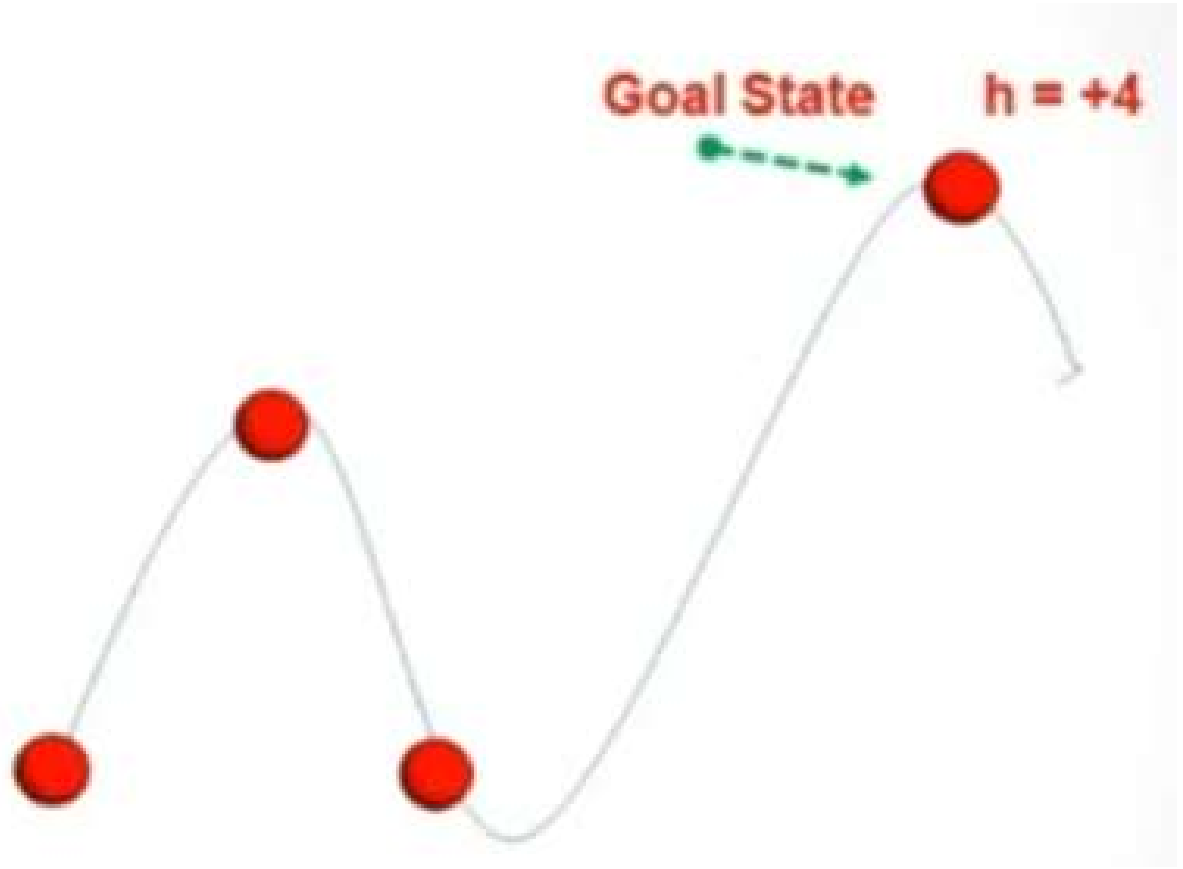









h = 0
h = +2
h = 0
h = 0


Goal State
h = +4

Blocks World Problem with Global Heuristic Function

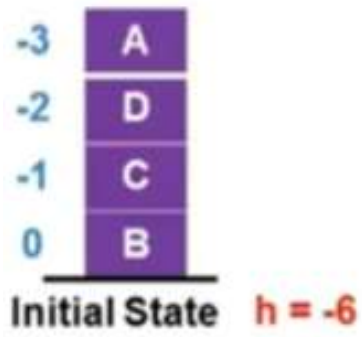
- *For each block that has the correct support structure : +1 to every block in the support structure.*
- *For each block that has the wrong support structure : -1 to every block in the support structure.*

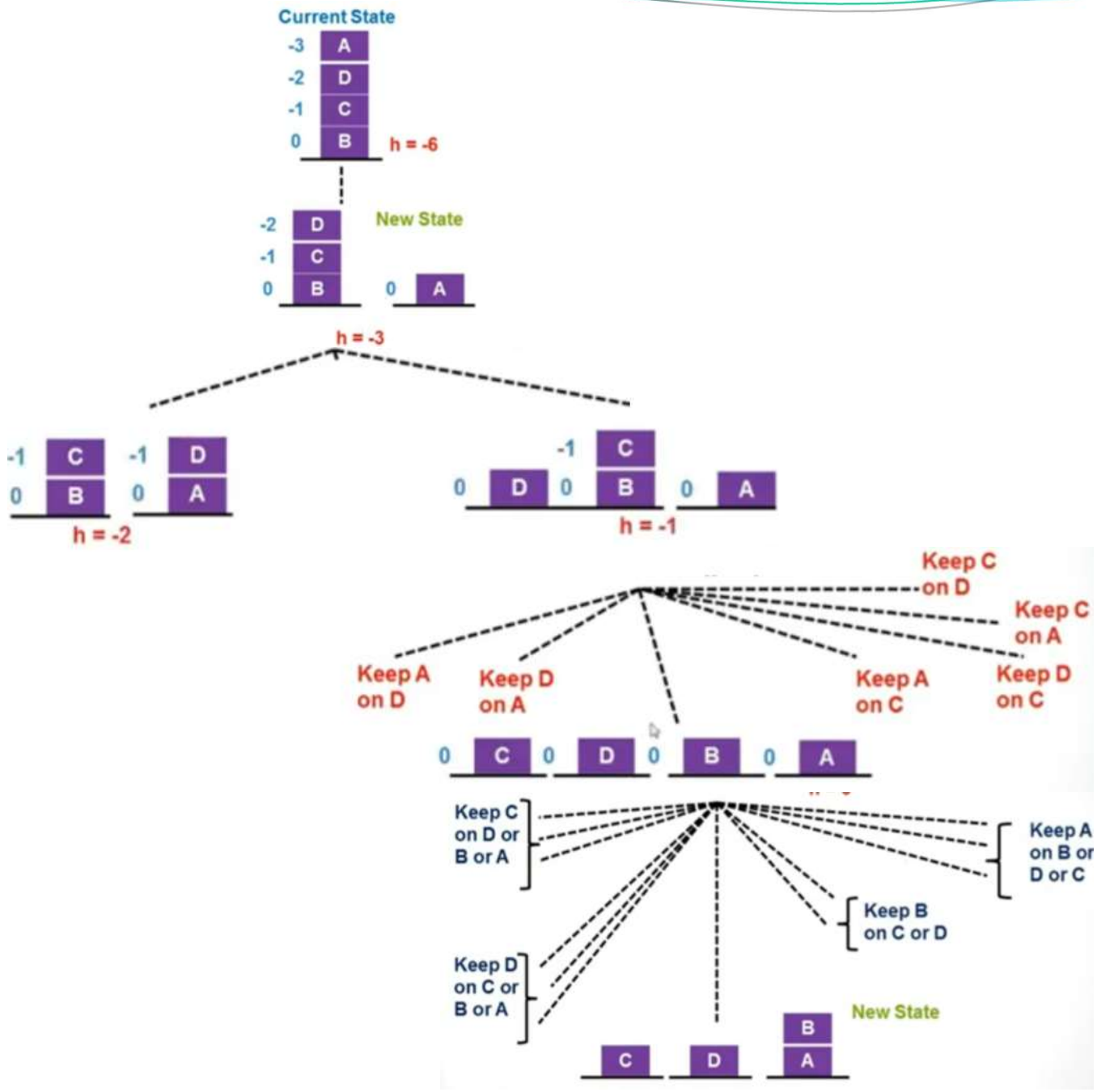
A
D
C
B

Initial

D
C
B
A

Goal




Current State
-3 A
-2 D
-1 C
0 B
h = -6
New State
-2 D
-1 C
0 B
0 A
h = -3
-1 C
-1 D
0 B
0 A
h = -2
-1 C
0 D
0 B
0 A
h = -1
Keep C on D
Keep C on A
Keep A on D
Keep D on A
Keep A on C
Keep D on C
0 C
0 D
0 B
0 A
Keep C on D or B or A
Keep A on B or D or C
Keep B on C or D
Keep D on C or B or A
New State
C
D
B
A

Blocks World Problem with Local Heuristic Function

+1 for each block that is resting on the thing it is suppose to be resting on.
-1 for each block that is resting on wrong thing.

A
H
G
F
E
D
C
B

Initial

H
G
F
E
D
C
B
A

Goal

Blocks World Problem with Global Heuristic Function

- ***For each block that has the correct support structure : +1 to every block in the support structure.***
- ***For each block that has the wrong support structure : -1 to every block in the support structure.***

A
H
G
F
E
D
C
B

Initial

H
G
F
E
D
C
B
A

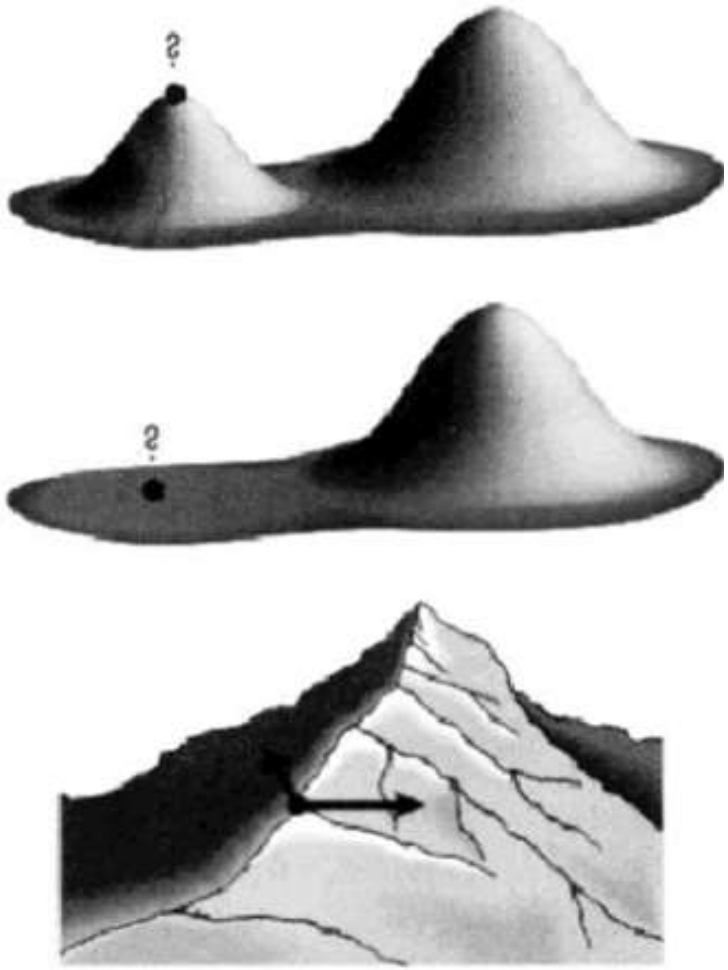
Goal

Solution to Overcome Problems in Hill Climbing

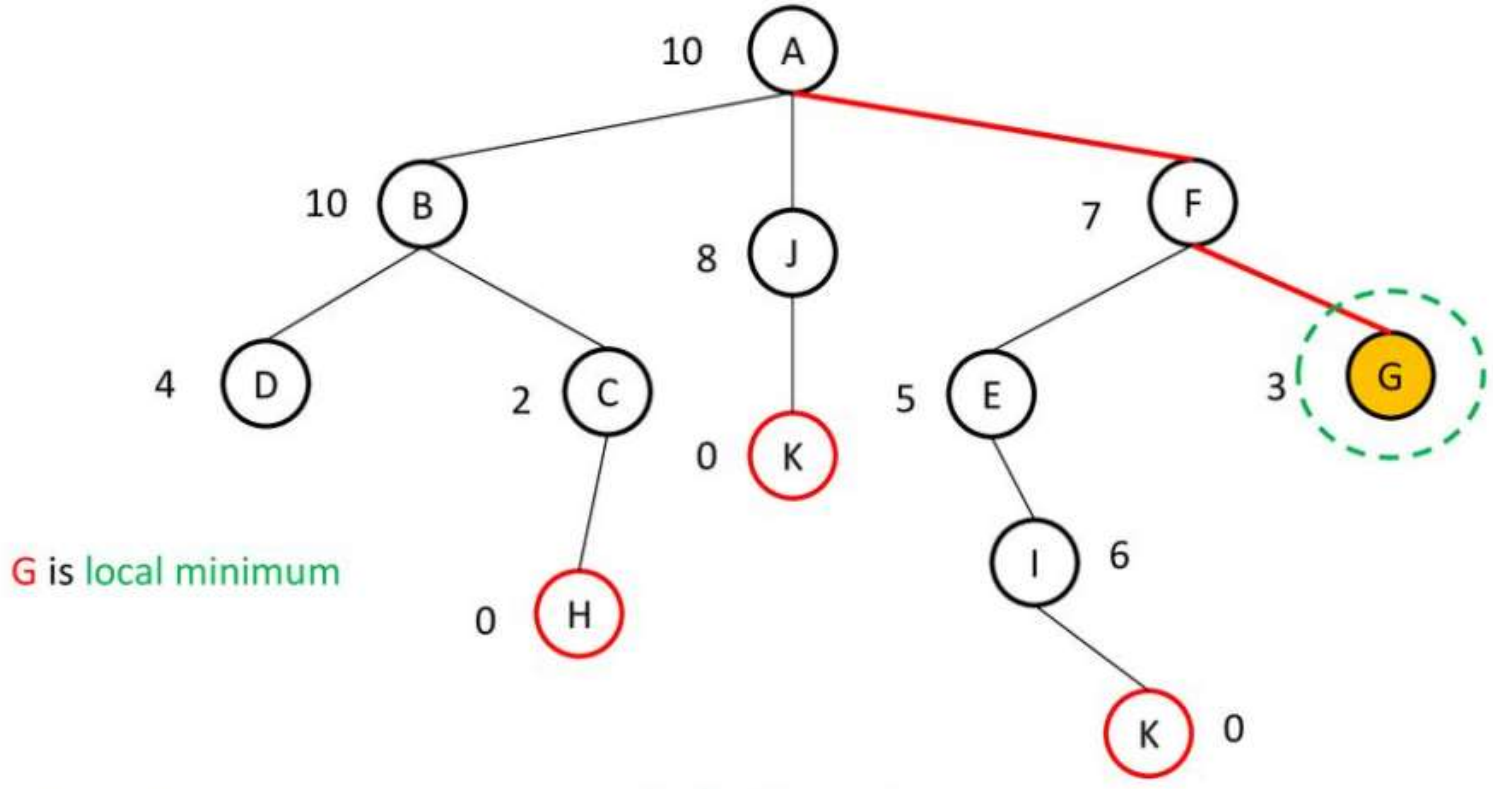
1. **Local maximum:**
   Utilize the ***backtracking* technique**. Maintain a list of visited states. If the search reaches an undesirable state, it can backtrack to the previous configuration and explore a new path.

2. **Plateau:** Make a **big jump**. Randomly select a state far away from the current state. Chances are that we will land in a non-plateau region.

3. **Ridge:** In this kind of obstacle, use **two or more rules** before testing. It implies moving in several directions at once.



Hill Climbing Search Example with Local Maxima



- Hill Climbing is sometimes called greedy local search because it grabs a good neighbor state without thinking ahead about where to go next.